

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

**INTEROPERABILITY AND SECURITY SUPPORT FOR
HETEROGENEOUS COTS/GOTS/LEGACY
COMPONENT-BASED ARCHITECTURE**

by

Tam M. Tran
James O. Allen

September 2000

Thesis Advisor:
Thesis Co-Advisor:

Luqi
Mantak Shing

Approved for public release; distribution is unlimited.

~~DATA QUALITY INSURANCE~~
**Reproduced From
Best Available Copy**

20001031 066

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
September 2000

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE

Interoperability and security support for heterogeneous COTS/GOTS/Legacy component-based architecture

5. FUNDING NUMBERS

6. AUTHOR(S)

Tran, Tam M. and Allen, James O.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release, distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

There is a need for Commercial-off-the-shelf (COTS), Government-off-the-shelf (GOTS) and legacy components to interoperate in a secure distributed computing environment in order to facilitate the development of evolving applications.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture.

A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM).

14. SUBJECT TERMS

COTS, GOTS, Application Wrapper, Security Model, Network Architecture, Component Interface, Open Standards

15. NUMBER OF PAGES
207

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT

Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT

Unclassified

20. LIMITATION OF ABSTRACT

UL

Approved for public release; distribution is
unlimited

**INTEROPERABILITY AND SECURITY SUPPORT FOR HETEROGENEOUS
COTS/GOTS/LEGACY COMPONENT-BASED ARCHITECTURE**

Tam M. Tran
B.S., San Diego State University, 1996

James O. Allen
B.A., University of California Los Angeles, 1970

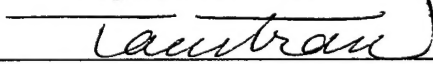
Submitted in partial fulfillment of the
Requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

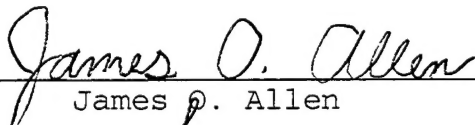
from the

**NAVAL POSTGRADUATE SCHOOL
September 2000**

Authors:

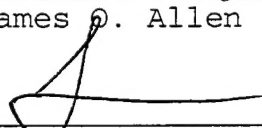


Tam M. Tran

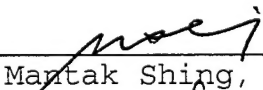


James O. Allen


Approved by:



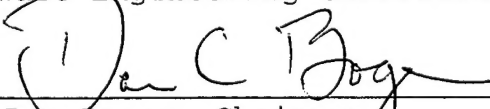
Luqi, Thesis Advisor



Mantak Shing, Thesis Co-Advisor



Luqi, Chairman
Software Engineering Curriculum



Dan Boger, Chairman
Department of Computer Science

ABSTRACT

There is a need for Commercial-off-the-shelf (COTS), Government-off-the-shelf (GOTS) and legacy components to interoperate in a secure distributed computing environment in order to facilitate the development of evolving applications.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture.

A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM).

TABLE OF CONTENTS

I. INTRODUCTION.....	1
II. EXISTING SOLUTIONS TO THE INTEROPERABILITY PROBLEM.....	5
A. GENERIC SECURITY SERVICE APPLICATION PROGRAM INTERFACE (GSS-API)	5
B. KERBEROS	5
C. A SECURE EUROPEAN SYSTEM FOR APPLICATIONS IN A MULTI-VENDOR ENVIRONMENT (SESAME)	7
D. DISTRIBUTED COMPUTING ENVIRONMENT (DCE)	7
E. KRYPTOKNIGHT.....	8
F. WINDOWS NT SECURITY MODEL	8
1. Local User Logon Process.....	9
2. Security Reference Monitor.....	9
3. Audit Security Subsystem.....	11
G. DCOM.....	12
H. JAVA	14
I. CORBA.....	15
J. SECURE SOCKETS LAYER (SSL)	16
K. SECURE HYPertext TRANSFER PROTOCOL (S-HTTP)	16
L. IP SECURITY (IPSEC).....	16
III. GENERIC WRAPPER FOR SYSTEM COMPONENTS.....	19
A. REQUIREMENTS OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS	19
1. General Description.....	19
2. Environment.....	21
B. SPECIFICATION OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS.....	22
1. XML Standard.....	23
a) Security	24
b) Namespaces.....	25
c) Document Type Definitions (DTDs)	25
d) Document Object Model (DOM).....	25
e) XML Specification.....	25
2. COTS Application exposes API.....	26
3. Standard file naming and directory conventions for component determination.....	28
4. Command line input support for COTS COMPONENTS Invocation....	29
IV. ARCHITECTURAL DESIGN PATTERN.....	31
A. ARCHITECTURAL DESIGN.....	31
B. NITES IMPLEMENTATION	32
1. Using Architectural Design Pattern.....	32
2. Thin Client Technology.....	33
3. Push Technology.....	33
C. NETWORK ARCHITECTURE	35
1. Intranet Security.....	37
2. Internet Security.....	37
3. Dial-in Security.....	37
V. CASE STUDY.....	39
A. CASE STUDY OVERVIEW.....	39
1. App.....	39
2. App Wrapper.....	39
3. System Monitor.....	40
4. System Controller.....	40
5. Storage Directory.....	40
6. Application (IMGEDT).....	40
7. Glue Component.....	40
8. Database.....	40
B. PRODUCE PRODUCTS TO DIRECTORY: IMAGE EDITOR (IMGEDT).....	43
C. DISPLAY PRODUCTS: CONTINUOUS BRIEF.....	46
1. Continuous Brief Initialization.....	47
2. Continuous Brief Update.....	50
3. User Interface.....	53

4. Brief Interfaces	53
a) Image Interface	53
b) Images Interface	54
c) Show Interface	54
D. DCOM DEPLOYMENT INSTRUCTIONS	55
VI. CONCLUSIONS	59
A. DCOM SOLUTION	59
B. ARCHITECTURAL DESIGN	59
C. WRAPPERS	60
D. SECURITY	61
E. IMGNT	61
F. FUTURE TRENDS	61
LIST OF REFERENCES	63
BIBLIOGRAPHY	65
APPENDIX A. GSS-API VERSION 2 FUNCTION CALLS	67
APPENDIX B. SESAME CRYPTOGRAPHIC SUPPORT FACILITY (CSF) APIS	69
INITIALIZATION APIS	69
RANDOM NUMBER GENERATION API	73
SET-UP AND CONFIGURATION	74
APPENDIX C. SESAME ARCHITECTURE	77
A. PROTOCOL NOTATIONS	77
B. USER SPONSOR FUNCTIONS	78
C. AUTHENTICATION PRIVILEGE ATTRIBUTE CLIENT (APA)	78
D. APPLICATION CLIENT	78
1. Authentication Server (AS) Functions	78
E. PRIVILEGE ATTRIBUTE SERVER (PAS) FUNCTIONS	79
F. KEY DISTRIBUTION SERVER (KDS)	79
G. PRIVILEGE ACCOUNT CERTIFICATE (PAC) VALIDATION FACILITY (PVF) FUNCTIONS	79
H. PUBLIC KEY MANAGEMENT (PKM) FUNCTIONS	80
APPENDIX D. SKELETON VB CODE FOR DESIGN PATTERN	81
A. MONITOR COMPONENT	81
1. Modules	81
a. Module 1	81
2. Classes	81
a. Monitor	81
b. Monitor Connector	82
B. CONTROLLER COMPONENT	83
1. Modules	83
a. Module 1	83
2. Classes	83
a. Controller	83
b. Controller Connector	84
C. GLUE COMPONENT	85
1. Classes	85
a. Glue	85
D. APPLICATION WRAPPER COMPONENT	85
1. Forms	85
APPENDIX E. XML VOCABULARIES	87
APPENDIX F. SYSTEMS REQUIREMENTS SPECIFICATION	93
1. SCOPE	95
1.1 INTRODUCTION	95
1.2 PURPOSE	95
1.3 BACKGROUND	96

1.4 REFERENCES	97
2. GENERAL DESCRIPTION.....	98
2.1 ARCHITECTURE GOALS	98
INTEROPERABILITY	98
ADOPTED FRAMEWORK TECHNOLOGY	99
SECURITY	100
NETWORK SECURITY	102
NETWORK COMMUNICATIONS	103
DEVELOPMENT LANGUAGE	103
2.2 ASSUMPTIONS AND DEPENDENCIES	103
3. TARGET ARCHITECTURE FUNCTIONS.....	105
SECURITY	105
GRAPHICAL USER INTERFACE (GUI)	106
EXTERNAL SYSTEM INTERFACES	106
MIDDLEWARE TECHNOLOGY	106
4. ARCHITECTURE ATTRIBUTES.....	108
4.1 PERFORMANCE REQUIREMENTS	108
4.2 RELIABILITY REQUIREMENTS	108
4.3 DESIGN CONSTRAINTS	109
APPENDIX G. SYSTEM DESIGN SPECIFICATION.....	111
1. SYSTEM ARCHITECTURE.....	112
1.1 SYSTEM ARCHITECTURE DIAGRAM	112
1.2 INTER-TASK COMMUNICATION	117
MONITOR/CONTROLLER.....	117
CONTROLLER/GLUE COMPONENT	118
CBWRAPPER/CONTROLLER.....	118
CBWRAPPER/GLUE COMPONENT	118
2. SUBSYSTEM DESCRIPTION.....	119
MONITOR	119
CONTROLLER	119
GLUE COMPONENT	119
CBWRAPPER	119
INITIALIZATION GUI	120
CONFIGURATION GUI	121
NAMING CONVENTION.....	121
THIN CLIENT TECHNOLOGY	122
PUSH TECHNOLOGY	122
Figure 3 - Wrapper & Glue Code Object Diagram	123
OMF	124
Table 1-11. OMF Attributes for the TAF Element	147
APPENDIX H. VISUAL BASIC IMPLEMENTATION.....	159
1. Configuration GUI (CBcfg).....	159
2. Application Wrapper (CBWrapper).....	166
3. Object Components (Continuous Brief).....	179
a) Global Variable Declarations	179
b) Timer.....	180
c) Controller	181
d) Controller Connector.....	185
e) Monitor	186
f) Monitor Connector	192
g) Glue.....	193
INITIAL DISTRIBUTION LIST.....	195

ACKNOWLEDGEMENT

The authors wishes to thank
Dr. Valdis Berzins and Dr. Mantak Shing
for their guidance in this project.

I. INTRODUCTION

There is a need for Commercial-off-the-shelf (COTS), Government-off-the-shelf (GOTS) and legacy components to inter-operate in a secure distributed computing environment in order to facilitate the development of evolving applications.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture.

A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM). The System Requirement Specification (SRS), System Design Specification (SDS) and Visual Basic Implementation, found in the appendices, are the results of a collaborative effort with graduate students Karen Gee and Thomas Nguyen.

Unified Modeling Language (UML) methodology is used in the formal specification of the system.

The Joint C4ISR Battle Center (JBC) Study considered several approaches to solving the interoperability problem, including wrappers, messaging, data mediators, data replicators, data translators, and ORBs, and evaluated each approach using the following criteria: performance, reliability, speed to field, cost, extendibility, COTS support, security and standards. The empirical scores for each criterion of each approach are plotted on a Kiviat graph. The JBC Study, published at the Naval Post Graduate School in 1999, recommends a solution in the context of ORBs, but with caveats. Re-evaluation is needed, as new products are available. Background and training of personnel is an important consideration in selecting a solution. [Ref. 1] This thesis also recommends the ORB approach and focuses on Microsoft Distributed Component Object Model (DCOM) with emphasis on setting security policy external to the component. Legacy applications are made DCOM compliant by wrapping the application within a DCOM component. Custom applications wrappers need to be designed, which is consistent with the findings of the JBC study.

This thesis is organized into the following chapters:

- Chapter II researches existing solutions to the distributed component integration problem.
- Chapter III proposes a methodology that can be used to transform desktop legacy applications into distributed web based applications.
- Chapter IV presents a design pattern application framework encompassing security and wrappers that is applied to the case study.
- Chapter V discusses the portion of the NITES system used as case study to validate the usefulness of the proposed methodology.
- Chapter VI presents the lessons learned and conclusions from the case study.

THIS PAGE IS INTENTIONALLY LEFT BLANK

II. EXISTING SOLUTIONS TO THE INTEROPERABILITY PROBLEM

A. GENERIC SECURITY SERVICE APPLICATION PROGRAM INTERFACE (GSS-API)

GSS-API is emerging as an Internet standard for securing applications. GSS-API is embedded in Common Object Request Broker Architecture (CORBA), Kerberos, Distributed Computing Environment/Remote Procedure Call (DCE/RPC), Sequence Packet Exchange (SPX), KryptoKnight, and SOCKS [Ref. 2]. GSS-API is popular because it is an interface specification that is independent of implementation mechanism, independent of placement, and independent of communication protocol. The interface specification is a product of the IETF Common Authentication Technology Working Group. Version 2 of GSS-API has 37 function calls broken down into 4 categories: Credential Management, context-level, per-message and support.

GSS-API assumes the application establishes a connection to a service, messages are transferred to and from the service, and the service will not request another external service on behalf of the user.[Ref. 2]

B. KERBEROS

Kerberos was developed in the 1980's at MIT to provide additional security for the Athena system. The primary goals were to provide single logon to a network of

application servers and protect authentication from masquerading attacks. Kerberos is an implementation mechanism for GSS-API. Kerberos assumes the client, network and server cannot be trusted and that a third party key distribution center (KDC) is needed to store secret keys. The KDC is composed of two logical entities, the authentication server (AS) and the ticket-granting server (TGS). The AS is responsible for authenticating the user and providing the user a ticket to access the TGS. The user sends its identity, server and nonce. A nonce is a randomly generated one-time value that is used to counter a replay attack. The AS responds with a session key, server and nonce encrypted using the user's secret key and a ticket encrypted with the server's secret key. The TGS is responsible for granting the user a ticket to access the requested server for a limited period of time. The user sends to the server an authenticator encrypted with the session key and the ticket obtained from TGS. The server decrypts the ticket to obtain the session key which in turn is used to decrypt the authenticator. Typically the authenticator has a timestamp that must be within 5 minutes of the current time. To provide mutual authentication the server returns the authenticator encrypted with the session key. Strong authentication is achieved because secret keys were never passed in the clear. [Ref. 3]

Kerberos has several weaknesses. The user's secret key is stored in the host's memory during AS exchange. Kerberos is vulnerable to password guessing attacks. Registering each service with the KDC does not scale. Applications must be modified to take advantage of Kerberos.

C. A SECURE EUROPEAN SYSTEM FOR APPLICATIONS IN A MULTI-VENDOR ENVIRONMENT (SESAME)

Sesame is the European substitute for Kerberos. Sesame implements all the specified security services. Sesame architecture can be divided into 4 major entities: client, security server, application server and support components. GSS-API calls need to be added to the client and application server entities in places where messages are being sent and received. The C source code for Sesame V4 for Redhat Linux V5 is available at www.cosic.east.kuleuven.ac.be/sesame. There is a project underway to convert Sesame to Java in order to improve portability.[Ref. 2]

D. DISTRIBUTED COMPUTING ENVIRONMENT (DCE)

The Open Systems Foundation (OSF) specification for DCE includes facilities for security, directory services, time services, threads and remote procedure calls.

DCE 1.2 is compatible with Kerberos V5 so single logon and mutual authentication services are available. DCE uses Access Control Lists (ACLs) for authorization. Role based authorization is not available. Like Kerberos, DCE/RPC uses

a session key to provide secure communication services between the client and server. A rich set of APIs, including GSS-API is available to the programmer. These APIs provide data confidentiality and integrity services.[Ref. 2]

The DCE web site is www.camb.opengroup.org/tech/dce.

E. KRYPTOKNIGHT

KryptoKnight has been under development at IBM since 1992. Kerberos influenced the design of this system. Similar security services include single logon per user, mutual authentication, key distribution and data integrity and confidentiality. Role based authorization is not provided. The 2-party, 3-party and inter-domain protocols are designed to minimize network usage and computer processing.[Ref. 2]

The KryptoKnight web page is www.zurich.ibm.com/~sti/g-kk/extern/kryptoknight

F. WINDOWS NT SECURITY MODEL

The goal of any multitasking and networked operating system security is to ensure that system resources such as memory, files, devices and CPUs cannot be accessed without authorization.

The NT security model has three major components: the logon process, the security reference monitor, and other security subsystems.

1. Local User Logon Process

Each user has an account on a local machine that is managed by administrators using the Security Accounts Manager (SAM). In a NT server environment, each user may also have a domain account. The Primary Domain Controller (PDC) and the Backup Domain Controller (BDC) are responsible for authenticating the user. Once authenticated, the user has access to any machine on the network that allows access to domain users. The trusted domain relationship is one-way and not transitive.

Each user may be assigned to one or more groups. If the number of users exceeds the number of groups, assigning users to groups and privileges and permissions to groups reduces the administrator's task of managing security policy.

2. Security Reference Monitor

The reference monitor is responsible for authorizing access to any NT object and audit generation. The reference monitor accesses all NT objects consistently and uniformly. User mode processes pass an object handle to system services operating in kernel mode.

There are 23 NT object types: adapter, controller, desktop, device, directory, driver, event, eventPair, file, IOCompletion, key, mutant, port, process, profile, section, semaphore, symbolicLink, thread,

timer, token, type, and windowStation. Each object type has a set of attributes that are common to all object types and a set of attributes specific to the object type. The object manager uses the common attributes to provide the following services: close, duplicate, query object, query security, set security, wait for single object, wait for multiple objects.

Each NT object has a security descriptor attribute which defines the permissions, auditing and ownership of an object. The corresponding structures are named Discretionary Access Control List (DACL), System Access Control List (SACL), and Owner Security Ids (OwnerSID). Each entry in the list is named an Access Control Entry (ACE). The owner controls a DACL ACE. The security administrator controls a SACL ACE. An ACE can contain a collection of access rights that may be generic, standard or specific. Generic access rights are read, write, execute and all (read, write, execute). Generic access rights can be mapped to standard access rights that are delete access, read access to security descriptor, read, write, execute, synchronize, write DAC, write Owner, required, and all.

In summary a user access token includes a Security ID (SID), a list of privileges and a list of group SIDs. An object security descriptor includes an owner SID, DACL, and SACL.[Ref. 4]

3. Audit Security Subsystem

The following table describes the types of events that can be audited in Windows NT.[Ref. 5]

Type of event	Description
Logon and Logoff	A user logged on or off or made a network connection.
File and Object Access	A user opened a directory or a file that is set for auditing in File Manager, or a user sent a print job to a printer that is set for auditing in Print Manager.
Use of User Rights	A user used a user right (except those rights related to logon and logoff).
User and Group Management	A user account or group was created, changed, or deleted. A user account was renamed, disabled, or enabled; or a password was set or changed.
Security Policy Changes	A change was made to the User Rights, Audit, or Trust Relationships policies.
Restart, Shutdown, and System	A user restarted or shut down the computer, or an event has occurred that affects system security or the security log.
Process Tracking	These events provided detailed tracking information for things like program activation, some forms of handle duplication, indirect object accesses, and process exit.

Table 1.1 Windows NT Event Types for Audit

The Event Viewer utility formats and displays audit event records.

Audit event records include header information that is present in all event records. The following list describes this common information.

- The time the event was generated.
- The SID of the subject that caused the event to be generated. If possible, Event Viewer translates this SID to an account name for display. The SID is the impersonation ID if the subject is impersonating a client, or the primary ID if the subject is not impersonating.
- The name of the system component or module that submitted the event. For security audits this is always Security.
- The module-specific ID of the specific event.
- The event type, either Success Audit or Failure Audit.
- The event category, used to group related events such as logon audits, object access audits, and policy change audits.[Ref. 5]

G. DCOM

Figure 1.1 shows the overall DCOM architecture. The client uses an interface, represented by a lollipop, to access a service provided by a remote component. Using DCE RPC and common security providers makes DCOM available on

other platforms including Apple Macintosh, Sun Solaris, Linux, AIX, and MVS.

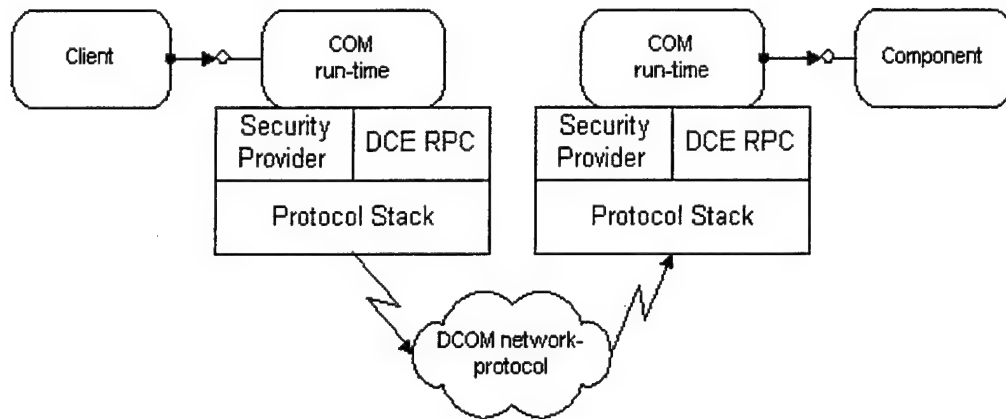


Figure 1.1. Overall DCOM Architecture [Ref. 5]

DCOM can provide security services for COTS components externally by using the DCOM configuration tool or by embedding security API calls within components. The primary DCOM security services fall into three categories: access, launch and call. Access security checks for privilege to connect to a running object. Launch security checks for privilege to create an object. Call security checks for privilege to access a component interface.

Each client has a security context that encapsulates security services. Security features, such as mutual authentication, can be selected just by setting a property value.

DCOM can impersonate the client on a server machine to allow nested client-server architecture. Impersonation can also be used to control access to individual properties and methods of components.

DCOM is layered on Object Remote Procedure Call (ORPC) which is an extension of DCE RPC. These services are accessible through the WIN32 Security Support Provider Interface (SSPI). DCOM can also accommodate multiple third party security providers.

DCOM uses Windows NT NTLM, Kerberos V5 or Distributed Password Authentication (DPA) authentication protocols.

DCOM uses SSL/PCT protocols to provide integrity and confidentiality services for communication connections.

DCOM uses the Windows Registry and the ACL facilities of the Windows NT operating system. DCOM is also available on Macintosh and UNIX platforms.[Ref. 4]

H. JAVA

Java 1.1 applets run in a virtual machine on a host machine. The assumption is that all applets are un-trusted unless accompanied by a digital signature. The virtual machine protects the host from un-trusted applets utilizing the "sandbox" approach. This means the capabilities of Java applications that are potentially harmful to the host are restricted in applets. For example, an applet may not access the host file system.

The `java.lang.SecurityManager` class implements the applet security restrictions. A security policy is created by instantiating and registering a security manager object. A potentially harmful operation causes an exception that is handled by a security manager method.

I. CORBA

The Common Object Services specification (CORBASec) describes security related tasks and requirements needed for CORBA.

A CORBA ORB, ORBacus, from Object Oriented Concept Inc. has been used to implement some specified security services. ORBacus currently provides the Security Level 1 functionality of CORBASec. Security Level 1 provides security services for applications that are unaware of security including mutual authentication, confidentiality and integrity.

The messages exchanged are encapsulated in the Secure Inter-ORB Protocol (SECIOP) message format. SECIOP provides a standard for maintaining security and interoperability between ORBs. Each end maintains its state following the rules of the SECIOP Context Management finite state machine.

The security functionality underneath is that of Kerberos V5 and is accessed through a Java binding of the GSS-API.

J. SECURE SOCKETS LAYER (SSL)

SSL is positioned between the TCP/IP application and connections layers enabling multiple services such as Telnet, HTTP and FTP to establish secure connections without modification to the services. SSL utilizes RSA Public/Private key architecture. The server identity is validated to the client by x.509 digital certificates. Optionally the client identity can also be validated to the server. The server has access to an LDAP compliant key directory server.[Ref. 6]

K. SECURE HYPERTEXT TRANSFER PROTOCOL (S-HTTP)

S-HTTP permits parties to negotiate symmetric or asymmetric keys, key management technique, message formats, and cryptographic strength. S-HTTP allows for multiple trust models to be negotiated between client and server. Security features are specific to the HTTP protocol.[Ref. 3]

L. IP SECURITY (IPSEC)

IPSec provides for secure transfer of IP packets across an untrusted network. IPSec resides at the network layer of the OSI model. IPSec is transparent to protocols at higher layers in the OSI model. IPSec is an open standard for encryption on an IP network.

Two one-way security associations (SA) between hosts or gateways store security parameters (Source IP, cryptographic algorithm, cryptographic keys, user or gateway name, data

sensitivity level, transport layer protocol, source and destination ports). Unique SA key includes security parameter index (SPI), IP destination, and security protocol, either Association Header (AH) or Encapsulated Security Payload (ESP). With ESP, the enclosed packet(tunneling) is encrypted, so original source and destination addresses could be unregistered.[Ref. 7]

THIS PAGE IS INTENTIONALLY LEFT BLANK

III. GENERIC WRAPPER FOR SYSTEM COMPONENTS

A. REQUIREMENTS OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS

1. General Description

The security services designed for commercial applications often focus on data integrity while military applications focus on data confidentiality. In order for COTS components to operate in a military environment, the commercial security services must be carefully selected to achieve military security requirements. The next section contains a list of security services applicable to the military environment that are also available in various combinations within commercial products. A methodology shall be developed to transform classes of legacy modules into reusable components using the wrapper architecture.

Components shall pass messages transparently across language, operating systems and network boundaries.

A common set of security services across operating systems will simplify implementation of a security policy.

The following security services shall be available to the customer:

- Single logon for users

- Mutual authentication
- Auditing
- Key distribution
- Role based Access Control
- Data confidentiality
- Data integrity
- Data availability
- Non-repudiation

The single logon for users means the user needs to identify him once per session. It is the responsibility of the security services to protect and distributed the authentication information of a user.

Mutual authentication ensures proper identification of the user to the system and the system to the user.

Auditing means significant security events are recorded for later analysis. Significant security events shall include login, logout, password change, and access validation.

Key distribution provides a secure transport mechanism for encryption keys.

Role based access control assigns roles to users and privileges to roles, thereby simplifying access control if the number of roles is less than the number of users.

Data confidentiality means data is disclosed according to a policy.

Data integrity means the recipient gets the intended data.

Data availability means the user has access to the data when needed.

Non-repudiation means the sender of a message cannot later deny he sent the message.

2. Environment

The classes of projects targeted by this thesis typically operate in an environment with the following conditions:

- Components pass messages synchronously or asynchronously.
- Components may have real-time constraints.
- A hierarchy of interacting COTS, GOTS and custom components may be assembled to form an application.
- Implementation will be dependent on the security services of the host operating systems.
- Security policies need to evolve and policy implementations need to be manageable in a distributed computing environment.

- Some components may be in binary executable form where compile or link is not possible. Other components may be re-linked but not recompiled. Other components may not be re-linked but substitution of dynamic load libraries (DLL) is possible. Other components may be modified at the source code level and recompiled.
- The security services will not be exported outside of the United States.
- Attacks can come from inside or outside an organization.
- This security system must be adaptable to counter new kinds of security attacks.
- The target systems will operate at a single level of security at no higher than the discretionary access control level (C2).

B. SPECIFICATION OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS

Wrappers that need to exchange self-describing content over a network can use XML. Utilization of XML within wrappers makes data transport mechanism independent of language or operating system. Following is a description of the XML standard.

1. XML Standard

XML is an emerging standard for transferring data among distributed components in web applications. Industry has been quick to agree on XML vocabularies. NITES has developed a nationally recognized vocabulary for meteorological data. See Appendix E for XML meteorological vocabulary and sources for other vocabularies.

XML offers the following desirable features:

- XML describes data that can be specified in a lexical tree structure. Unlike directed graphs, trees can be efficiently traversed.
- XML and HTML share the same level in the WEB architecture. Both can use the secure HTML mechanism and the digital signature mechanism.
- XML specification is the product of the World Wide Web Consortium (W3C) and is recognized as a standard for distribution of data over the Internet.
- All content is encoded in the specified Unicode character set. There is no need to wrap vendor specific data formats.
- Industry specific XML vocabularies make content available to any compliant application.

- XML vocabularies are extensible without affecting earlier versions.

Any DoD joint application should consider evolving to XML. Some common steps to gradually incorporate XML into an existing project include:

- Categorize the types of information the system handles. Examples are personnel, weather, tactical, and logistics.
- Search for existing XML standards in categories.
- If there are no XML standards within a category, organize a standards committee, and produce an industry wide standard.
- Develop components to transform existing messages, records, etc. into XML entities. A one-time transformation is usually preferable to repeated run-time transformations.
- Use existing tools to provide additional transformations such as record set to XML.
- Use security zones of the browser to implement security policy. Use XML parser imbedded in browser to extract information for presentation.

a) Security

The security zone features have been extended in Internet Explorer 5 (IE5) to provide security services

for the embedded XML parser. The zones include local, Internet, local intranet, trusted site, and restricted site in order of trustworthiness. The originating zone may access a zone that is equal or less trustworthy.[Ref. 5]

b) Namespaces

XML namespace specification developed by World Wide Web Consortium (W3C) is implemented on IE5. This allows developers to define unique element names using a registered qualifier.

c) Document Type Definitions (DTDs)

DTDs utilize XML to describe rules to validate an XML document. DTDs are an optional section of the XML document.

d) Document Object Model (DOM)

The DOM provides a standard way to programmatically construct and traverse any XML document. The XML document is composed of objects with attributes and methods. DOM can be applied to the task of transforming an ActiveX Data Object (ADO) record set into an XML document. Interfaces are defined for the DOM and all XML objects.

e) XML Specification

The XML specification is on the Web at URL www.w3.org/xml. Production rules are in the Extended

Backus-Naur Format (EBNF). An annotated version is at Web site www.xml.com/xml/pub/axml/axmlintro.html.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

[Ref. 8]

2. COTS Application exposes API

DCOM and CORBA use an Interface Definition Language (IDL) to name and describe an interface containing public attributes, methods and events. There is a many-to-many relationship between interfaces and

components. A component may implement one or more interfaces. The interface serves as a contract between the component developer and user.

How do you ensure each interface has a unique name when many independent activities are creating interfaces? One solution is to use a routine that will always generate a different name each time it is called. DCOM uses this solution to generate unique class and interface names. Once an interface has been assigned a name it will never change. There is no way to modify an interface and use its original name. This guarantees that all legacy code will never need to be changed because an interface has been modified.

DCOM interfaces are language and platform independent. For example, a component written in Visual Basic and running on a Windows NT platform can use a component written in C++ and running on a Unix platform.

DCOM and CORBA require each component to implement the Unknown interface. From this interface, all interfaces implemented by the component can be dynamically discovered.

Dynamic discovery and use of an interface is known as late binding. Use of a priori knowledge of implemented interfaces is known as early binding. DCOM and CORBA both support early and late binding. There is a performance penalty for using late binding.

Microsoft Visual Basic hides many interface details. The development environment generates the IDL from the class implementation. The unique IDL name is automatically generated. The clause "with events" will enable receipt of events. The Unknown interface is automatically generated.

Microsoft Word, Excel and Powerpoint are examples of COTS components that expose an API. In the case study the Powerpoint API is used by the application wrapper.

3. Standard file naming and directory conventions for component determination

On Windows NT there is a many-to-one relationship between a file type and an application. For example, the file type PPT is associated with the PowerPoint application.

NITES imagery applications generate TIF, GIF, and MIF file types. PowerPoint is capable of processing the above file types.

Middleware wrappers can take advantage of standard file naming conventions and directory conventions to integrate components. For example, if a COTS application periodically generates an imagery file to a known directory, middleware can poll the directory for new files with a file type of interest and pass the file to a consumer of the file type.

4. Command line input support for COTS COMPONENTS Invocation

UNIX and DOS have popularized starting an application and passing switches and parameters on a command line. This same mechanism can be used from within a program to start another program. A wrapper can use this mechanism to integrate independent COTS applications.

A chaining model is used when the calling program terminates after execution. An asynchronous model is used when the calling and called programs operate in parallel. A synchronous model is used when the calling program waits for completion of the called program.

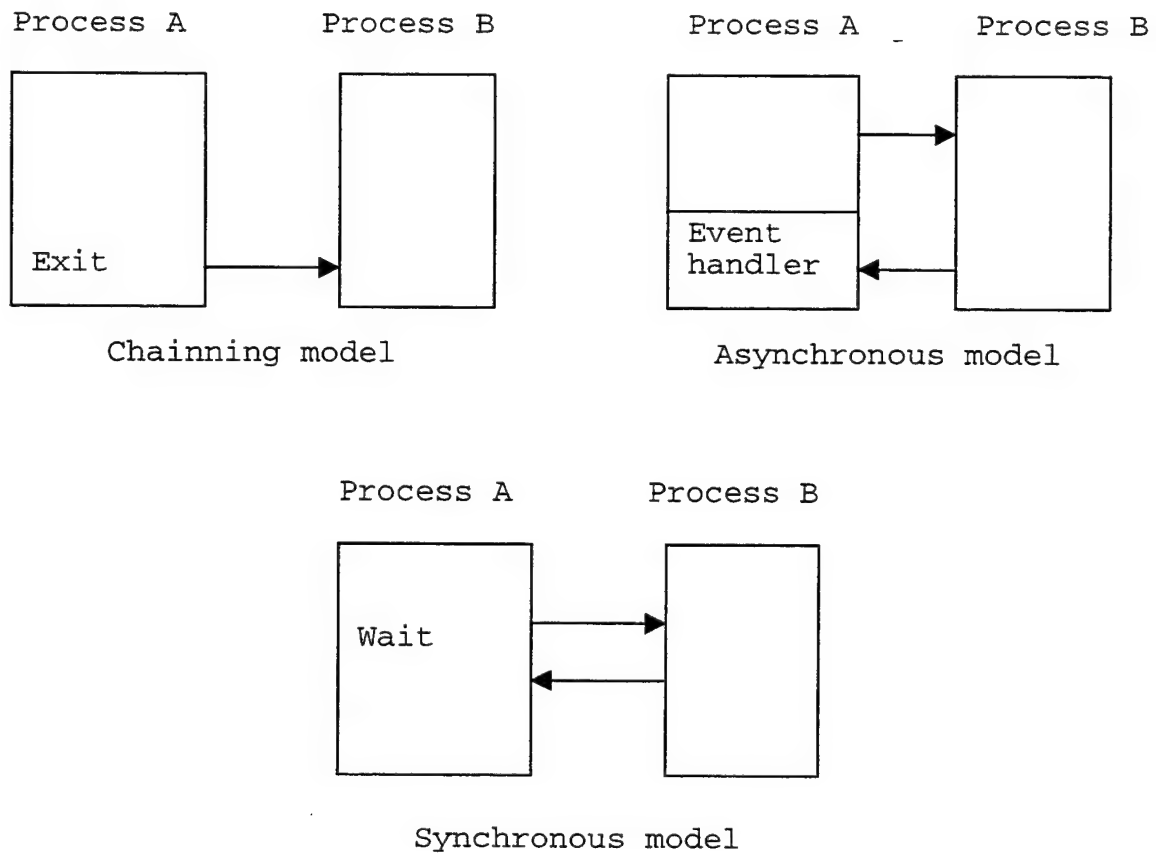


Figure 3.1. Wrapper calling models

IV. ARCHITECTURAL DESIGN PATTERN

A. ARCHITECTURAL DESIGN

The architectural design pattern represented in Figure 4.1 is common to many IT systems including NITES and USCG National Distress Response System Modernization Program (NDRSMP).

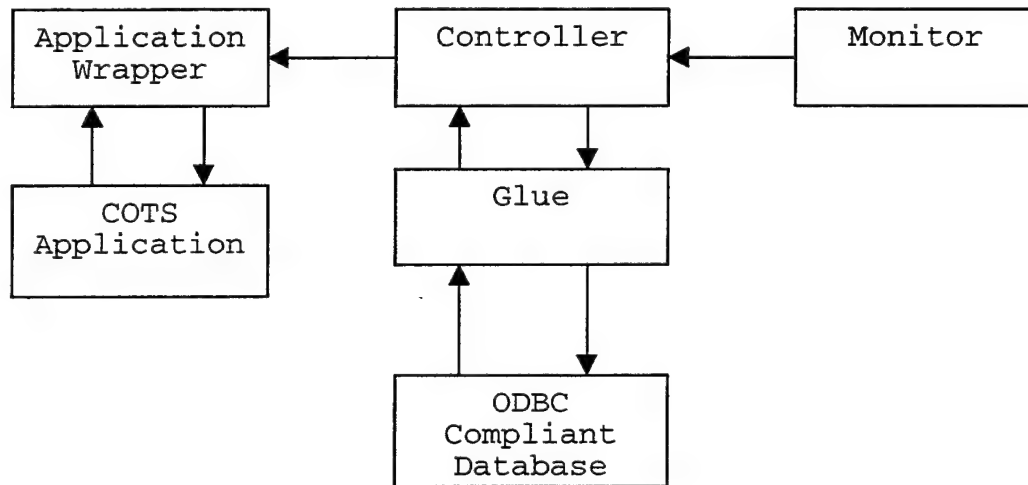


Figure 4.1. Architectural Design Pattern

The realization of this architecture on a network of Windows NT machines running DCOM, IIS, Internet Explorer and optionally a UNIX relational database server machine, satisfies the requirements of the previous section.

In NITES, the object is a TIF file containing a satellite image. In NDRSMP, the object is a WAV file

containing a voice segment. The Monitor component is responsible for detecting the presence of a new object. The controller component is responsible for coordinating multiple concurrent asynchronous activities. The glue component is responsible for storing and retrieving objects from a ODBC compliant relational database. The Application Wrapper is responsible for making the object available to a COTS viewer application.

B. NITES IMPLEMENTATION

1. Using Architectural Design Pattern

A Windows NT DCOM solution in Visual Basic (VB) was used in NITES to implement the architectural design pattern. See Appendix D for the skeleton VB code. The launch, access and permission security features were set external to each component using DCOMCNFG utility. The DCOMCNFG utility was also used to set the location of each component and user account assigned to the component. The automation data types were used to make marshaling and un-marshaling of data transparent to each component. Migration from a desktop application to an Internet Explorer 5 (IE) was performed to reduce maintenance. Client components can be maintained on the server and automatically downloaded to the client. Migration is accomplished by

converting the project type from standard executable to an ActiveX control using Microsoft Visual Studio.

The key to generic wrapper design is to use standard objects. Standard objects include widely used file extensions such as Tagged Image File Format (TIFF) and WAV, XML meta data, and record sets. There are COTS plug-in viewers for each of the above standard object types.

2. Thin Client Technology

The web based application wrapper is implemented using modern thin client technology. When a user opens a HTTP page from a browser, the wrapper is then automatically downloaded and installed on the client machine. Once the wrapper is up and running, all images needed for creating the brief are dynamically downloaded from the server using the OpenURL method. OpenURL uses the current open HTTP connection to transfer image files. The continuous brief is created on the client machine using the PowerPoint APIs. The PowerPoint is used to display the brief.

3. Push Technology

The advantage of using push technology is that the client does not need to poll the server periodically for new data. The server notifies its clients (wrapper) when new data (images) arrive. The wrapper receives the notification and compares the image type with the type

being showed. If the image types match, the wrapper downloads a new set of images from the server and updates the brief.

C. NETWORK ARCHITECTURE

Figure 4.2 depicts network architecture similar to many systems including NITES. The network is composed of an intranet divided into four sub-nets, a router connecting the four sub-nets and providing a connection to the internet service provider, and a dial-in access server. Two sub-nets separate the traffic of two user groups. Security and packet wrapper options within this network architecture are characterized. The components in the architectural design pattern are typically deployed on the web server and user computers.

Subnets

Router

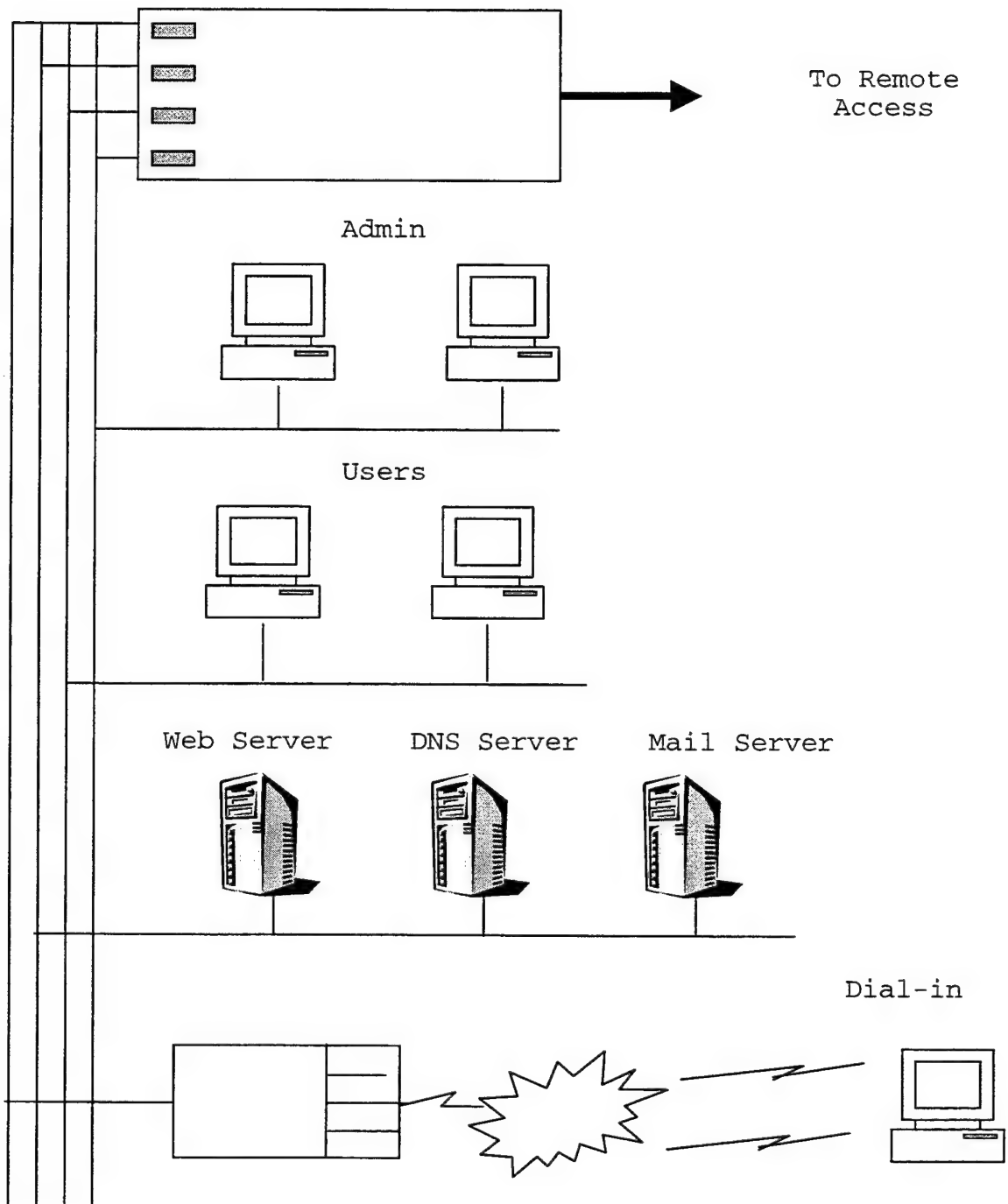


Figure 4.2. Network Architecture

1. Intranet Security

A hierarchical network architecture formed with routers offers traffic isolation and additional security. Using ACLs and IP filters on the router Ethernet interfaces can control traffic flow across subnets. Some routers, including the popular Cisco router, are capable of protecting against IP spoofing.

2. Internet Security

Standard security mechanisms are available at different layers of the OSI Network Model. Point-to-point tunneling protocol (PPTP), Layer 2 tunneling protocol (L2TP), Frame Relay, and Asynchronous transfer mode (ATM) are available at the Data link layer. IP security (IPSec) and Generic routing encapsulation (GRE) are available at the Network layer. SOCKSv5, SSL and TLS are available at the session layer.

3. Dial-in Security

Some authentication schemes, such as password authentication protocol (PAP), transfer passwords in the clear and are vulnerable to snooping. Stronger authentication schemes are available.

The dial-in access server is a convenient place to host authentication schemes for mobile users. Remote Authentication Dial-in User Service (RADIUS) is a draft standard that covers protocols for a centralized access

server. RADIUS allows for one-time token authentication schemes.

Windows NT provides Challenge Handshake Authentication Protocol (CHAP). Client and server share a common secret key. A unique session key is negotiated without transferring the secret key in the clear. A unique session key limits the usefulness of replay attacks to the current session.

V. CASE STUDY

A. CASE STUDY OVERVIEW

A subset of the operational NITES system was chosen for the case study. This subset is representative of the issues involved in the integration of COTS software components where only the executables are available.

The case study covers the wrapper and security aspects of component integration.

The wrapper transforms COTS applications into a COM/DCOM component enabling interfaces with infrastructure components as shown in Figure 5.2.

1. App

The App is the COTS application that provides the APIs used by the App Wrapper to integrate with other components.

2. App Wrapper

The App Wrapper is the software code developed to add, modify, and hide functionality from COTS, GOTS or legacy software components to align them with the overall system requirements and architecture. In the design, wrapper and glue code technology is being implemented to enable the COTS applications to adhere to the existing NITES architecture.

3. System Monitor

The Monitor component is responsible for detecting the presence of a new object.

4. System Controller

The controller component is responsible for coordinating multiple concurrent asynchronous activities. The controller runs on the application server. It serves two functions within the system, handling notifications from the monitor and the glue component.

5. Storage Directory

The Storage Directory is a target directory that is accessed by the IMGEDT application and the Glue component. This is the location for the data temporarily stored before being updated to, or retrieved from the database.

6. Application (IMGEDT)

IMGEDT is a COTS application that generates the satellite images.

7. Glue Component

The glue component is responsible for storing and retrieving objects from an ODBC compliant relational database.

8. Database

The Database is an ODBC compliant relational database that is available for storing and retrieving data.

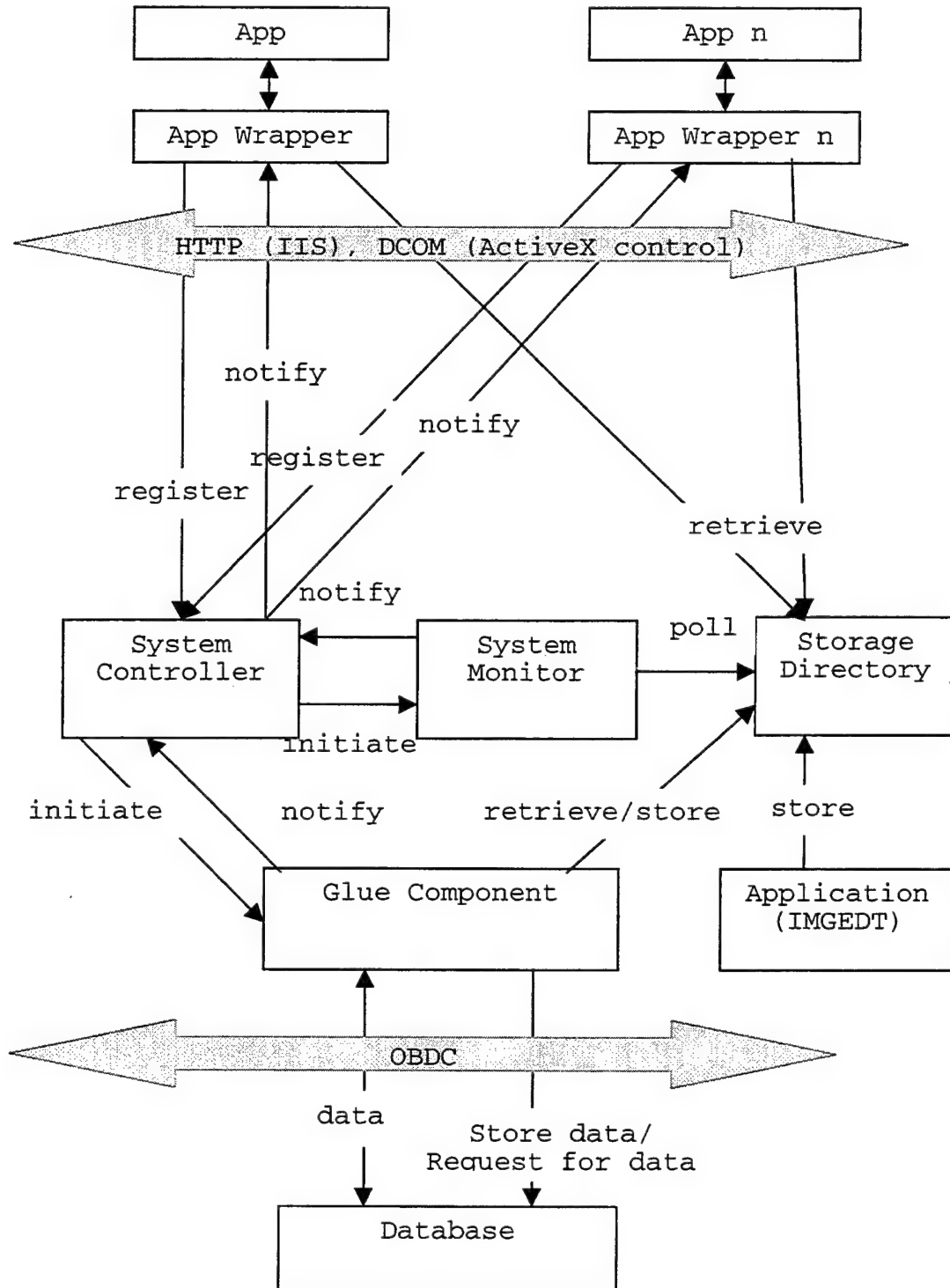


Figure 5.2 Component Integration DCOM Wrappers

Component security is based on external DCOM security features. External DCOM security provides the following advantages over internal DCOM security:

- Source code, object code or DLLs are not required. External security can be used when only executables are available.
- Since security policy is not embedded within components, components may be reused in security environments.
- Security policy can be implemented without writing any code or understanding component internals.

The case study focuses on two COTS applications within the operational NITES system. The first application, called image editor, produces a product. The second application, called continuous brief, presents a product. The image editor creates a file in a known directory. The file extension identifies the file type. The file is saved in a central relational database. This conforms to a design philosophy of NITES that each application interfaces with the database and not with each other.

The continuous brief loops through a set of the latest weather satellite images. The satellite images are extracted from the database. Continuous brief parameters include the

number of images, viewing duration of each image, and image viewing dimensions.

Each application fits the three-tiered architecture of presentation, logic, and database. The presentation and logic tiers run on a PC with Windows NT. The database tier runs on Sun Solaris. COM/DCOM is used to interface logic components on the PC. ADO/ODBC is used to interface to the relational database.

The Extensible Markup Language (XML) is used to wrap the data products in the relational database.

B. PRODUCE PRODUCTS TO DIRECTORY: IMAGE EDITOR (IMGEDT)

IMGEDT is a legacy NITES application that will be used to demonstrate the effectiveness of the design pattern produce products to directory. It is assumed only the executable is available, dynamic link library (DLL) substitution is not an option, and driver chaining will not be used.

IMGEDT is a Windows NT desktop application with no network or database connectivity. IMGEDT is capable of opening an image file, editing an image file and saving an image file to the local directory system.

The user signs on locally using id and password. The user has system privileges and object permissions to execute IMGEDT, read an image file and store an image file to a

directory. Windows NT provides authentication and access control services.

Figure 5.3 shows the product producer sequence diagram. It is the responsibility of the System Monitor to poll the IMGEDT target directory for new or updated image files. It is assumed the IMGEDT target directory is located on a shared drive within an intranet and that the shared drive is accessible to the System Monitor. When a file is detected, the System Monitor initiates the sequence to store the image on a remote relational database.

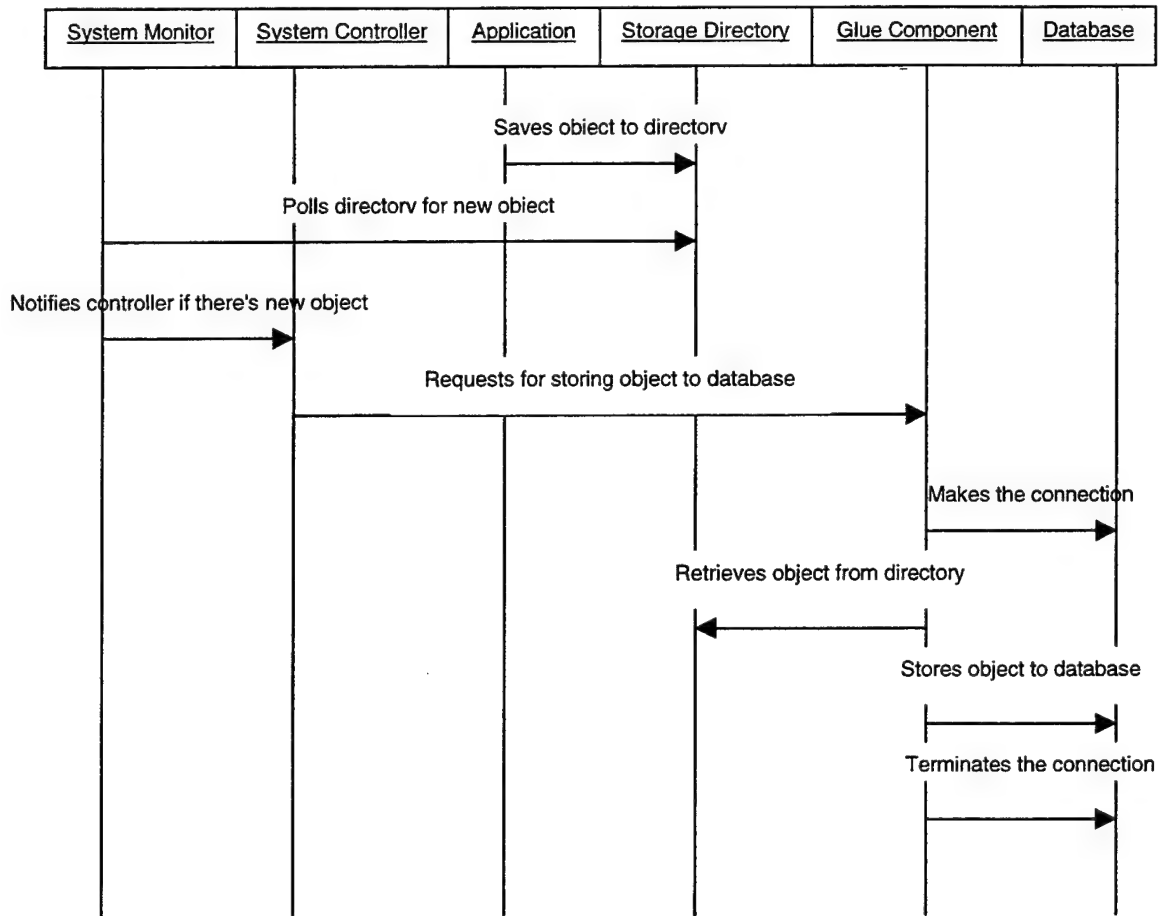


Figure 5.3 Store object into Database

Following is a detailed explanation of each step in the sequence diagram.

1. The application saves an object to the storage directory.
2. Concurrent to step 1, the system monitor periodically polls the storage directory for a new or updated object.

3. Access to the object is allowed only if the system monitor has read permission.
4. The system monitor notifies the system controller if there is new object.
5. The glue component establishes a remote connection to the relational database.
6. The glue component updates the database.
7. The relational database commits the object to the database after the command is successfully processed.
8. The glue component terminates the remote connection to the relational database.

C. DISPLAY PRODUCTS: CONTINUOUS BRIEF

The goals of the continuous brief case study are:

1. Prove that the presented wrapper and security architecture is feasible in the context of an existing system.
2. Measure performance impact due to security and wrappers.
3. Formalize the case study into a pattern for future projects.

The continuous brief is composed of the following objects:

1. Web Browser
2. PowerPoint as an ActiveX Document embedded within a browser.

3. PowerPoint Application wrapper that utilizes PowerPoint API.
4. Control that coordinates activities within the system
5. Communications that provide inter-component messaging facilities.
6. Database that provides storage and retrieval of row sets using SQL.
7. IMGNT application that interfaces with the database for storing and retrieving images.

1. Continuous Brief Initialization

Figure 5.4 shows the sequence of actions performed by cooperating objects to initialize the continuous brief.

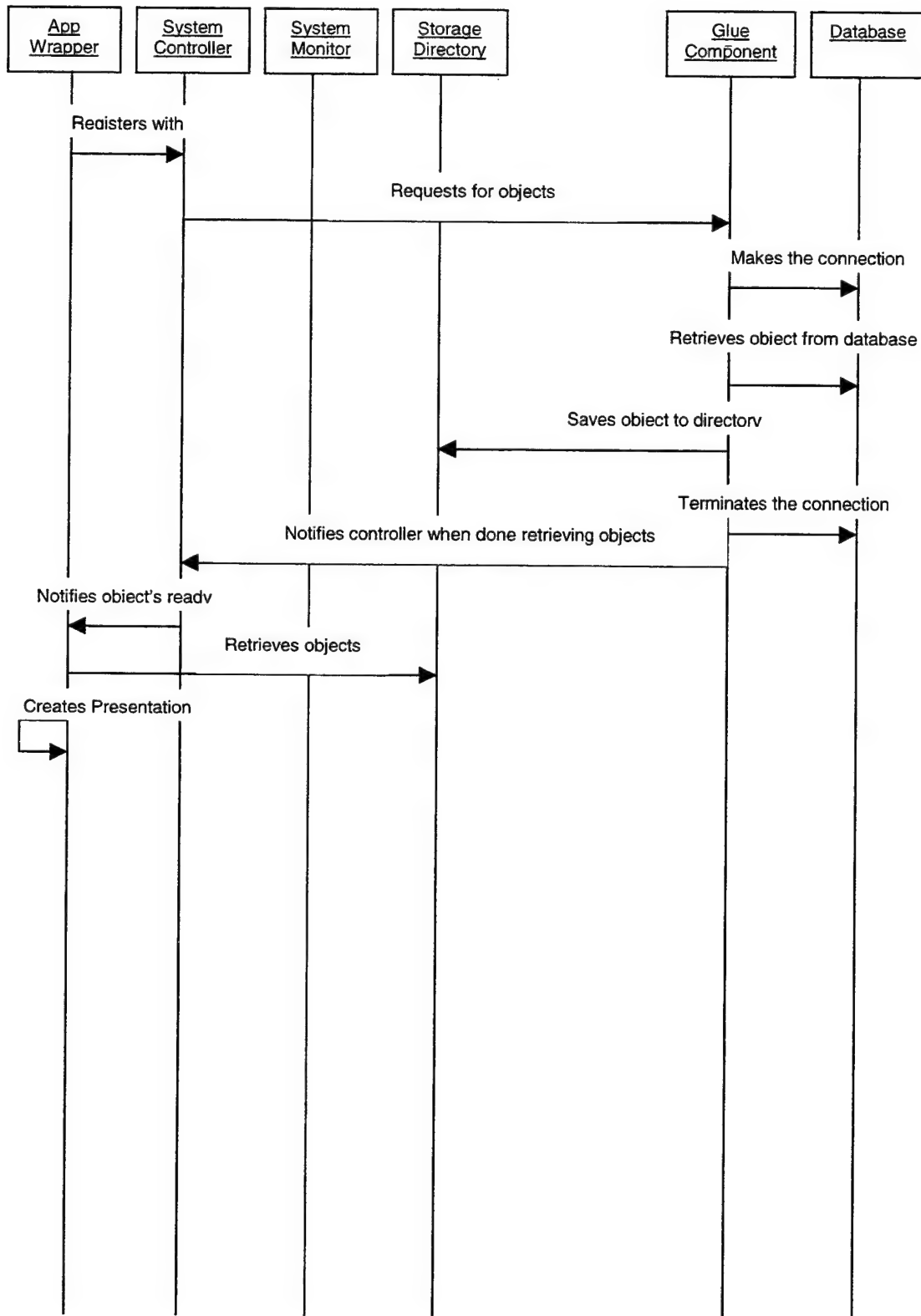


Figure 5.4. Continuous Brief Initialization Sequence Diagram

Following is a description of the diagram:

1. User registers to the web server. User authentication scheme will depend on user role and user location.
2. If user is authenticated, the web server sends the Initialization GUI home page containing parameters to be filled in.
3. The user fills in the number of images starting from the most current, the display duration of each image in seconds and the height and width of the display area. Default values are 24 images, 0 second duration, and display area equal to the screen size.
4. The web Server initiates the application wrapper and passes input parameters.
5. The application wrapper registers interest in new satellite images with the controller. The controller will notify all registered application wrappers when a new satellite image has been stored into the database.
6. The application wrapper requests the latest requested number of images from the database.

7. The glue component transforms the request into an asynchronous database query.
8. The database returns the requested images in a tif, jpeg or mif file format. The time the satellite image was photographed is part of the file name.
9. The glue component saves the requested images to the storage directory.
10. The application wrapper downloads the images via the current HTTP connection.
11. The application wrapper uses the PPT API to generate and show a continuous brief.

2. Continuous Brief Update

Figure 5.5 shows the sequence of actions performed by cooperating objects to update the continuous brief.

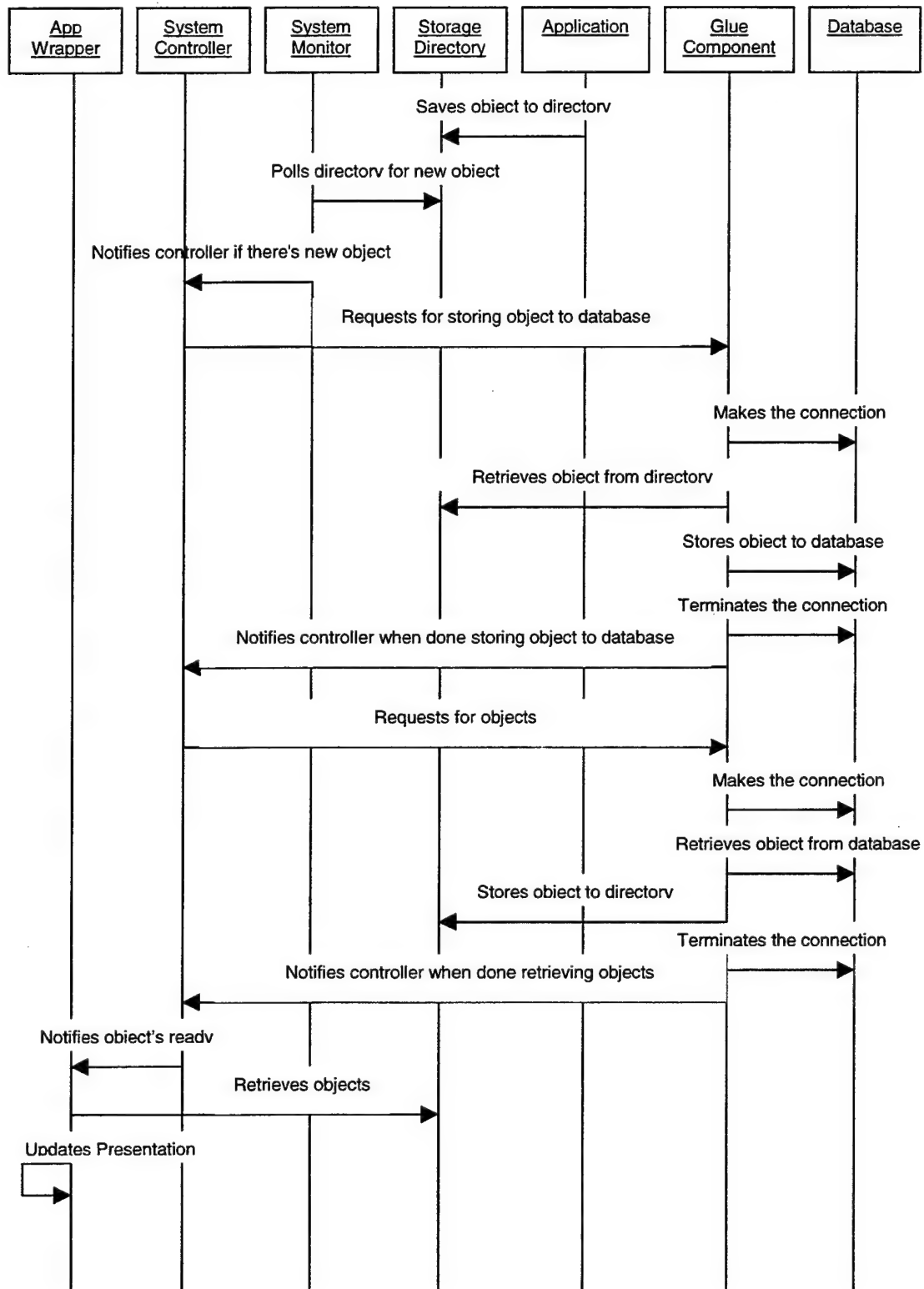


Figure 5.5 Continuous Brief Update Sequence Diagram

It is assumed that the App wrapper is embedded in the browser on the client machine. Following is a description of the diagram:

1. The Application saves new object to the storage directory.
2. The system monitor notifies system controller there is new object.
3. Controller forwards request to Glue component.
4. Glue component marshals request for database query and sends request using ODBC protocol.
5. Database processes request and stores the new object.
6. Glue component notifies controller that a new object has been inserted into the database.
7. System controller requests Glue component for objects.
8. Glue component initiates retrieval of objects from database.
9. Glue component notifies system controller when retrieval is completed.
10. Controller notifies registered App wrappers that new objects are available.
11. App wrapper updates presentation with new objects.

The Observer Pattern, as described in *Design Patterns*, also classifies this type of application. The subject is the satellite image section of the database and the

observer is the application wrapper. The loose coupling between the database and the wrapper allows multiple wrappers to receive notification of a new satellite image.

3. User Interface

Before the brief is started, the user is prompted for the following parameters:

- The type of brief. Default is visual.
- Number of images in brief (1-99). Default 24
- Duration of each image (0-20 seconds). Default 0.
- Image display dimensions (height and width in twips). Default is window size.

These parameters initialize the brief via the brief interfaces. Buttons are used to start and stop the brief. A reset button restores input parameters to default values.

4. Brief Interfaces

a) Image Interface

The image interface is mapped to the PowerPoint shape object interface. Each image in the brief share the following properties:

SetWidth (twips width);

Sets the width of the display area in twips
for the image.

SetHeight (twips height);

Sets the height of the display area in twips
for the image.

Each image is sized to fit the display area.

b) Images Interface

The images interface is mapped to the PowerPoint slides
object interface. The interface manages the images in
the brief.

SetNumberOfImages (integer nImages);

Sets the number of images in the brief.

AddImage (picture image);

Adds the given image to the end of the brief.

The images should be added in time sequence
from the oldest to the newest.

c) Show Interface

The show interface is mapped to the PowerPoint show
object interface. The interface manages the sequential
display of each image in the brief.

SetImageDuration (integer seconds);

Sets the number of seconds that each slide is
displayed.

StartShow ();

Display images from first to last and repeat
image sequence until show is stopped.

StopShow ();

Stop continuous brief.

D. DCOM DEPLOYMENT INSTRUCTIONS

The Visual Basic development environment provides tools to create a deployment package for ActiveX Exe remote servers. The remote server check box inside the project/properties/component section needs to be checked. Making the project using Files/Make creates an executable file (EXE), assigns a globally unique class ids and interfaces ids, and registers the component on the local machine. To avoid creation of new global identifiers each time the component is made, set the version compatibility to binary compatibility using the projects/properties/component pane. New global identifiers are only necessary when the interface definition changes. The package and deployment wizard steps you through the process of creating a deployment package. Since the target machine does not usually contain a development environment, the Visual Basic run time environment must be included in the deployment package. If the remote server component creates other components, the Visual Basic Reference file (VBR) and Type

Library (TLB) must also be included in the deployment package.

Transfer the deployment package to the target machine and execute the setup application. Setup will register the component in the registry, copy dependent files to the appropriate system directory and update the programs folder. Run DCOMCNFG on the server machine. The DCOM server check box needs to be checked in order for the DCOM server to run. Find the application name from the list of applications, and select properties. The location is local machine. The security setting controls user roles that have privileges to launch, attach or change ownership of the remote server. The identification section is used to enter the user account and user password that will be used to launch the component. The protocol section is used to list the protocols to use in priority sequence.

Run DCONCNFG on the client machine. The DCOM server check box needs to be checked in order for the DCOM server to run. Find the server application name from the list of applications, and select properties. The location is the name of the remote server machine. The security setting controls user roles that have privileges to launch, attach or change ownership of the client component. The identification section is used to enter the user account and user password that will be used to launch the component.

The protocol section is used to list the protocols to use in priority sequence.

The client is now ready to launch or attach to the remote server component. There is no need to manually start the server component. When the client creates a new the server component, the server component is launched on the remote machine.

Use the internet package option of the Package and Deployment Wizard to deploy an ActiveX control to the Web Server. This creates a CAB file containing the control and its dependencies. The CAB file is compressed to reduce download time. During the initial download, the ActiveX control is saved and registered on the client. Subsequent references to the control are resolved locally.

THIS PAGE IS INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

The following conclusions are based on application of the distributed component integration methodology (DCIM) to the case study.

A. DCOM SOLUTION

DCOM is a natural choice for this implementation. The host machine is a PC running Windows NT and DCOM is bundled with the OS. There is familiarity with DCOM from prior projects. Visual Basic development environment hides low-level plumbing from the developer. Security policy can be defined external to the component implementation. The existing design pattern template fit the design of the continuous brief application.

DCOM proved to be a quick and efficient way to implement a robust continuous brief application. Components were tested in the VB debug environment. Then executables were tested on a single machine. Finally, the system was distributed to the Web server machine. No source code changes were made to execute in these three configurations.

B. ARCHITECTURAL DESIGN

The architectural design with accompanying VB application framework skeleton code proved to simplify implementation. The details of object creation, push technology, client registration for service, event

processing, browser based components, asynchronous object execution, and polling were provided by the framework.

The framework was extended to poll a directory, make asynchronous database queries, add arguments to events, wrap PowerPoint and add a user interface. The developer is able to focus on the application without being distracted by plumbing details.

C. WRAPPERS

Three types of wrappers were used in the implementation of the continuous brief: file type in directory, object, and COTS API. The monitor component of the architectural design was extended to periodically check for a new satellite image file in a directory specified by the configuration utility. The object wrapper used the file name structure to extract image time, type and location. The PowerPoint API was used show the continuous brief. Even though the show could have been easily implemented using a Java applet, PowerPoint could simplify future extensions such as image cropping and image titling.

To eliminate the need for PowerPoint on each client, the show could have been generated on the server and sent to the client for viewing. Microsoft provides a web based PowerPoint viewer free of charge.

D. SECURITY

The external security features of DCOM proved to simplify implementation of security policy; however Windows NT Service Pack 5 does not expose DCE encryption to external DCOM security. Single user logon, user privileges based on role and discretionary access control were available.

E. IMGNT

Administrative problems precluded the use of ImgNT to retrieve selected images from a database and store in a directory. The system had not been installed on an unclassified system, Visual Basic was not available, and ImgNT patches had not been made. It is assumed that ImgNT had already stored requested images to a directory.

F. FUTURE TRENDS

The value of the results of this thesis is time sensitive. Research on this thesis began in April 1999. Since that time Microsoft has released Windows 2000, SPAWAR has unveiled a public key infrastructure for e-mail, SPAWAR has a draft security policy, a network centric architecture has been deployed to the USS Coronado, CORBA has a wider selection of commercial ORBs, new standards for wireless communications have been developed, Linux is gaining support from many communities, security measures are receiving higher priority and many other innovations.

The distributed component integration methodology described in the thesis will remain in the mainstream for the foreseeable future. Independently designed components will need custom integration using some form of wrapper. Network administrators will require implementation of security policy using tools external to the application.

LIST OF REFERENCES

- [1] Berzins V., Luqi, Schultes B. *JBC Report*, Naval Post Graduate School, 1999
- [2] Ashley, P., *Practical Intranet Security*, Kluwer Academic Publishers, 1999
- [3] Summers Rita C., *Secure Computing*, McGraw-Hill, 1997
- [4] Grimes, R., *Professional DCOM Programming*, WROX, 1997
- [5] Microsoft Corporation, *Entire Collection*, MSDN Library, 1996
- [6] Krause M., *Handbook of Information Security Management*, Auerbach, 1999
- [7] Phaltankar K., *Implementing Secure Intranets and Extranets*, Artech House, 2000
- [8] Moulitis N., Kirk C., *XML Black Book*, Coriolis Technology Press, 1999
- [9] Szyperski, Clemens, *Component Software*, Addison-Wesley, 1998

THIS PAGE IS INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

Berzins and Luqi, *Software Engineering with Abstractions*,
Addison-Wesley, 1991

Douglas B., *Real-Time UML*, Addison-Wesley, 1998

Gamma E., Helm R., Johnson R., Vlissides J., *Design Patterns*
CD, Addison-Wesley, 1995

http://www.esat.kuleuven.ac.be/cosic/sesame3_2.html

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX A. GSS-API VERSION 2 FUNCTION CALLS

CREDENTIAL MANAGEMENT

GSS_Acquire_cred	acquire credentials for use.
GSS_Inquire_cred	display information about credentials.
GSS_Release_cred	release credentials after use.

CONTEXT-LEVEL CALLS

GSS_Init_sec_context	initiate outbound security context.
GSS_Accept_sec_context	accept inbound security context
GSS_Delete_sec_context	flush context.
GSS_Process_context_token	process received control token on context.
GSS_Context_time	indicate validity time remaining in context.

PER-MESSAGE CALLS

GSS_GetMIC	apply signature, receive as token separate from message.
GSS_VerifyMIC	validate signature token along

	with message.
GSS_Wrap	sign, optionally encrypt and encapsulate.
GSS_Unwrap	decapsulate, decrypt if needed, validate signature.
SUPPORT CALLS	
GSS_Display_status	translate status codes to printable form.
GSS_Compare_name	compare two names for equality
GSS_Display_name	translate name to printable form.
GSS_Import_name	convert printable name to normalized form.
GSS_Release_name	free storage of normalized-form name.
GSS_Release_buffer	free storage of printable name
GSS_Release_oid_set	free storage of OID set object

APPENDIX B. SESAME CRYPTOGRAPHIC SUPPORT FACILITY (CSF) APIS

INITIALIZATION APIS

csf_get_qos()

Returns the list of allowed pairs of algorithms with associated key length, for a given quality of service, within a given CSF domain such as "quality of service". The first algorithm and key length pair represent the default.

A quality of service is

- A service (integrity or confidentiality),
- A strength (weak, medium or strong),
- A class of algorithms (symmetric or asymmetric)

csf_begin()

Starts CSF up for a given algorithm. This API is used to initialize internal data for a software algorithm, or to set-up a hardware device.

csf_end()

Turns off CSF for a given algorithm. This API is used to free internal data for a software algorithm, or to shut down a hardware device.

Key generation APIs

A key handle is generated by these APIs.

csf_gen_asym_key_pair()

Generates an asymmetric key pair with the key length, key data and the reversible cryptographic algorithm as parameters.

csf_gen_sym_key()

Generates a symmetric key with the key length, key data and the reversible cryptographic algorithm as parameters.

csf_derive_secret_key()

This API is used to derive a secret key of a given key length from a string or a basic key, using an irreversible encryption algorithm and a seed.

Key handling

csf_init_key()

Initializes the key to be used by the CSF module. An indication on the way the key is stored (hardware, software, smart card ...), on the way the key is used (encryption, decryption, signature key or a key to check a signature) and the key itself or a reference of that key is given in input. It returns an opaque key handle to be used by subsequent calls to CSF APIs.

csf_release_key()

Releases an opaque key handle.

csf_read_key_info()

Allows to retrieve a key or a key reference from a key handle.

csf_get_key_data()

Allows to retrieve key data (key usage and optionally key validity time, initial vector) from a key handle.

Crypto context APIs

csf_init_context()

Initializes a crypto context from a CSF key handle and a pair of algorithms (reversible or irreversible) and associated key length. This context contains elements (hardware or software) to be used in data protection operations. It returns an opaque context handle to be used by subsequent data protection CSF APIs.

If the crypto context already exists, it is modified according to the input parameters.

csf_create_owf_context()

Creates a CSF context, only usable for an irreversible encryption algorithm which does not use any key, such as MD4 or MD5. No key handle is needed to use this interface.

csf_release_context()

Releases an opaque CSF context handle.

csf_duplicate_context()

Duplicates an existing crypto context. A new context handle is generated. The new context can then be modified by a call to csf_init_context().

csf_retrieve_key_from_context()

Returns the key handle attached to a crypto context.

csf_query_context()

Returns the pair of algorithms (irreversible + reversible) with associated key length and the quality of service attached to a crypto context.

Data protection APIs

csf_encrypt()

Generates an encrypted text from a clear text and a crypto context (including a key, a reversible algorithm and optionally initial vectors).

csf_decrypt()

Generates a clear text from an encrypted text using a crypto context (including a key and a reversible algorithm).

csf_generate_check_value()

Generates a signature from a clear text using a crypto context (including a key (private or secret), an irreversible algorithm and a reversible one).

csf_verify_check_value()

Checks the signature of a clear text using a crypto context (including a key (public or secret), an irreversible algorithm and a reversible one).

csf_owf()

Generates an irreversibly encrypted text from a clear text using a crypto context (including an irreversible algorithm).

Import/export APIs

csf_extract_key()

Packs the key and all data relative to the key (key usage, key validity) into an exportable format. This package has to be sent to the remote machine. `csf_restore_key()` has then to be called on this machine to restore the key information.

csf_restore_key()

Creates a key handle from a package obtained by an earlier call to `csf_extract_key()`, usually on another machine.

csf_extract_context()

Packs the key and all data relative to the crypto context (key usage, key validity, pair of algorithms) into an exportable format. This package has to be sent to the remote machine. `csf_restore_context()` has then to be called on this machine to restore the context information.

csf_restore_context()

Creates a key handle from a package obtained by an earlier call to `csf_extract_key()`, usually on another machine.

RANDOM NUMBER GENERATION API

csf_gen_rand_num()

Generates a random number of a given length.

Free routines

free_key_info()

Free a key (A `key_info_t` structure).

free_key_data()

Free key data (a `key_data_t` structure).

free_algo_id ()

Free an algorithm (an `algo_identifier_t` structure).

free_algo_id_pair()

Free a pair of algorithms (an `algo_id_pair_t` structure).

free_algo_id_pair_list ()

Free a list of algorithms (an `algo_id_pair_list_t` structure).

free_algo_list_except_one()

Free a list of algorithms, except one pair in the list.

SET-UP AND CONFIGURATION

Set-up and configuration of the CSF module is done by a control program called `csfcp`.

The CSF administrator is the only person authorized to run this program.

`csfcp` is be used to:

- Configure the quality of service, within the local domain. A list of allowed pairs of algorithm identifiers (irreversible or reversible) is to be associated to each qos.
- Configure the quality of service which is to be used to communicate between two CSF domains. A subset of the local qos configuration can be chosen and then sent to the second domain.

- Set-up all the algorithms available under CSF. For all available algorithms, the choice between hardware and software is made, for key storage and algorithm implementation.

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX C. SESAME ARCHITECTURE -

A. PROTOCOL NOTATIONS

A	Authentication Server
P	Privilege Attribute Server
U	User Sponsor
R	User
X	Client Application
Y	Server Application
Z	Server Application accesses by delegate
V	PAC validation facility of application server Y
W	PAC validation facility of application server Z
K_{AB}	Long term key shared between A and B
k_{AB}	Session key shared between A and B
PK_A	Public key of A
PK_A^{-1}	Private key of A
$ReQPriv_R$	Requested privileges by user R sealed by k_{UP}
$Cert_i$	X.509 certificate for the public key Pk_i
RL_x	Requested lifetime for x
T_s, T_e	Start and end time
r_i	Nonce generated by i
n_i	Message sequence number
$h()$	Hash function
$KeyPK_{i-j-k} = ENC(PK_j)(k_{jk}, T_s, T_e, data)$	

$$\text{KeyPK}_{j-k} = \text{ENC}(\text{PK}_k)(k_{jk}, T_s, T_e, \text{data})$$
$$\text{AuthSK}_{i-j} = \text{ENC}(k_{ij})(j, t_i, \text{data})$$
$$\text{AuthPK}_{i-j} = \text{SIGN}(\text{PK}_i^{-1})(j, t_i, \text{KeyPK}_{i-j})$$

B. USER SPONSOR FUNCTIONS

- Sends an authenticator $\text{SIGN}(\text{PK}_R^{-1})(A, t_R, \text{Key}(\text{PK}_{R-A}))$ to the Authentication Server.
- Decrypts the incoming key package from AS using the user's private key.
- Sends a request for a PAC to the privilege attribute server. The request contains the requested lifetime of the PAC, TGT, session key authenticator $\text{ENC}(k_{u-p})(P, t_u, \text{data})$.

C. AUTHENTICATION PRIVILEGE ATTRIBUTE CLIENT (APA)

The APA is developed by a programmer using the GSS-API. The User Sponsor uses this API to communicate with the authentication server and privilege attribute server to obtain authentication and credentials. See Appendix A for a description of GSS-API.

D. APPLICATION CLIENT

Every application client needs to be modified to include GSS-API.

1. Authentication Server (AS) Functions

Checks the X.509 certificate for the public key of user (Cert_R).

Verifies the authenticator portions of Cert_R .

Returns an authentication which includes the Primary Principal Identifier (PPID) as part of the ticket granting ticket (TGT), and an authenticator containing the public key of the privilege attribute server (PAS)

$$\text{TGT}_R = \text{ENC}(K_{AP})(R, U, T_s, T_e, k_{UP})$$

$$\text{PAC}_R = \text{SIGN}(\text{PK}_P^{-1})(\text{user role attributes}, \text{PPID}_R, \text{PV}_R, \text{DTQ}_R, \text{data})$$

E. PRIVILEGE ATTRIBUTE SERVER (PAS) FUNCTIONS

Supplies PAC as specified in ECMA 219 Security in Open Systems, 2nd edition, March 1996. European Computer Manufacturers Association

F. KEY DISTRIBUTION SERVER (KDS)

- For the intra-domain case use Kerberos V5 model.
- For the inter-domain case use X.509 certificates.

G. PRIVILEGE ACCOUNT CERTIFICATE (PAC) VALIDATION FACILITY (PVF) FUNCTIONS

- Validate PAC
- Key Management

Support Components

- Audit
- Record security relevant events using appropriate identities.

H. PUBLIC KEY MANAGEMENT (PKM) FUNCTIONS

- Manage public and private keys using PGP solution
- Establish symmetric keys between parties *i* and *j* using public-key standard X.509.

i sends a session key to *j* encrypted with *j*'s public key. *i* sends an authenticator using its private key. *J* authenticates the message signature by applying *i*'s public key and comparing the message with the message signature. The session key is now available to both parties.

APPENDIX D. SKELETON VB CODE FOR DESIGN PATTERN

A. MONITOR COMPONENT

1. Modules

a. Module 1

```
Option Explicit
Public gMonitor As Monitor           ' Reference to monitor
Public glngUseCount As Long          ' Global reference count
```

2. Classes

a. Monitor

```
Option Explicit
```

```
Private mFormForTimer As FormForTimer
Private WithEvents mTimerForMonitor As Timer
```

```
Public Enum Enumeration
```

```
    enum1 = 1
    enum2 = 2
    enum3 = 3
```

```
End Enum
```

```
' Event that passes all automation data types supported by
' proxy and stub
```

```
Event MonitorActivity( _
    bool As Boolean, _
    chr As Byte, _
    sfloat As Single, _
    dfloat As Double, _
    sint As Integer, _
    lint As Long, _
    enum123 As Enumeration, _
    str As String, _
    money As Currency, _
    datetime As Date)
```

```
Private Sub Class_Initialize() ' Start Monitor Timer
```

```
    ' Create instance of form
    Set mFormForTimer = New FormForTimer
    Load mFormForTimer
```

```

    ' Connect timers' events to associated event procedures
    ' in Monitor
    Set mTimerForMonitor = mFormForTimer.TimerForMonitor

End Sub

Private Sub Class_Terminate()      ' Terminate Monitor
    Set mTimerForMonitor = Nothing
    Unload mFormForTimer
    Set mFormForTimer = Nothing
End Sub

Private Sub mTimerForMonitor_Timer()      ' Process Timer
Event

    Dim bool As Boolean
    Dim chr As Byte
    Dim sfloat As Single
    Dim dfloat As Double
    Dim sint As Integer
    Dim lint As Long
    Dim enum123 As Enumeration
    Dim str As String
    Dim money As Currency
    Dim datetime As Date

    '<insert monitor task>

    ' Signal clients that monitor has detected activity
    RaiseEvent MonitorActivity(bool, _
                                chr, _
                                sfloat, _
                                dfloat, _
                                sint, _
                                lint, _
                                enum123, _
                                str, _
                                money, _
                                datetime)

End Sub

```

b. Monitor Connector

```
Option Explicit
```

```
Public Property Get Monitor() As Monitor ' Get reference to
                                         ' monitor
```

```
        Set Monitor = gMonitor
End Property
```

```
Private Sub Class_Initialize() ' Create Monitor and
                                ' reference count
    If gMonitor Is Nothing Then
        Set gMonitor = New Monitor
    End If
    glngUseCount = glngUseCount + 1
End Sub
```

```
Private Sub Class_Terminate() ' Terminate Monitor when
                                ' reference count = 0
    glngUseCount = glngUseCount - 1
    If glngUseCount = 0 Then
        Set gMonitor = Nothing
    End If
End Sub
```

B. CONTROLLER COMPONENT

1. Modules

a. Module 1

```
Option Explicit
Public gController As Controller ' Reference to controller
Public glngUseCount As Long      ' Global reference count
```

2. Classes

a. Controller

```
Option Explicit

Event ControllerEvent() ' Sent to AppWrapper(s)

Public WithEvents mglue As Glue ' WithEvents causes glue to
                                ' run asynchronously
Private WithEvents mMonitor As Monitor ' Get Monitor events

' Multiple connections to single monitor
Private mMonitorConnector As MonitorConnector

Private Sub Class_Initialize() ' Connect to Monitor

    Set mMonitorConnector = New MonitorConnector
    Set mMonitor = mMonitorConnector.Monitor

End Sub
```



```

' Receive event from Monitor
Private Sub mMonitor_MonitorActivity( _
    bool As Boolean, _
    chr As Byte, _
    sfloat As Single, _
    dfloat As Double, _
    sint As Integer, _
    lint As Long, _
    enum123 As Enumeration, _
    str As String, _
    money As Currency, _
    datetime As Date)

    Set mglue = New Glue
    Call mglue.StartGlue                ' Glue runs
    asynchronously
End Sub

Private Sub mglue_glueDone()          ' Asynchronous glue
    component is done

Set mglue = Nothing
RaiseEvent ControllerEvent
End Sub

```

b. Controller Connector

```

Option Explicit

Public Property Get Controller() As Controller
    Set Controller = gController
End Property

Private Sub Class_Initialize()        ' Initialize Controller
    ' and reference count
    If gController Is Nothing Then
        Set gController = New Controller
    End If
    glngUseCount = glngUseCount + 1
End Sub

Private Sub Class_Terminate()         ' Terminate controller when
    reference count = 0
    glngUseCount = glngUseCount - 1
    If glngUseCount = 0 Then
        Set gController = Nothing
    End If
End Sub

```

C. GLUE COMPONENT

1. Classes

a. Glue

Option Explicit

Event GlueDone() ' Sent when glue task done

Public Sub StartGlue() ' Start glue task

 ' <Insert glue task here>

 RaiseEvent GlueDone

End Sub

D. APPLICATION WRAPPER COMPONENT

1. Forms

Option Explicit

Private WithEvents mController As Controller

Private mControllerConnector As ControllerConnector

Private Sub Form_Load() ' Connect to controller

 Set mControllerConnector = New ControllerConnector

 Set mController = mControllerConnector.Controller

End Sub

 ' Receive Controller event

Private Sub mController_ControllerEvent()

Text1.Text = "Received Controller Notification"

 ' <insert interface with COTS application>

End Sub

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX E. XML VOCABULARIES -

The following list contains sources for some existing XML vocabularies:

Mathematical Markup Language (MathML) can be found at URL www.w3.org/Math

Web Interface Definition Language (WIDL) can be found at URL www.webmethods.com/technology/widl_description.html

The Nites I Meteorological Vocabulary Observation Markup Format (OMF) :

```
<!-- <!DOCTYPE OMF SYSTEM "OMF.dtd" [ -->
<!-- Weather Observation Definition Format DTD -->
<!-- This is the OMF XML DTD. It can be referred to using
the
formal public identifier
-//METNET//OMF 1.0//EN
For description, see OMF.html
$id: OMF.dtd,v 3.8 1999/10/25 18:18:31 oleg Exp oleg $
-->
<!-- Weather Observation Definition Format -->
<!-- Basic attributes -->
<!ENTITY % TStamp-type "NMTOKEN">
<!ENTITY % TRange-type "CDATA">
<!ENTITY % TStamp "TStamp %TStamp-type; #REQUIRED">
<!ENTITY % TRange "TRange %TRange-type; #REQUIRED">
<!ENTITY % LatLon "LatLon CDATA #REQUIRED">
<!ENTITY % LatLons "LatLons CDATA #REQUIRED">
<!ENTITY % BBox-REQD "BBox CDATA #REQUIRED">
<!ENTITY % BBox-OPT "BBox CDATA #IMPLIED">
<!ENTITY % Bid "Bid NMTOKEN #REQUIRED">
<!ENTITY % SName "SName CDATA #REQUIRED">
<!ENTITY % Elev "Elev NMTOKEN #IMPLIED">
<!-- Basic elements -->
<!ELEMENT VALID (#PCDATA)>
<!ATTLIST VALID %TRange;>
<!-- A collection of weather observation reports -->
<!ELEMENT Reports ( METAR | SPECI | UAR | BTSC | SYN )*>
<!ATTLIST Reports %TStamp;>
```

```

<!-- Common report attributes -->
<!ENTITY % ReportAttrs
"%TStamp; %LatLon; %BId; %SName; %Elev;
Vis NMToken #IMPLIED
Ceiling NMToken #IMPLIED
">
<!-- METAR and SPECI reports -->
<!ELEMENT METAR (#PCDATA)>
<!ATTLIST METAR %ReportAttrs;>
<!ELEMENT SPECI (#PCDATA)>

<!ATTLIST SPECI %ReportAttrs;>
<!-- A collection of weather hazard advisories -->
<!ELEMENT Advisories ( SIGMET | AIRMET | WW )* >
<!ATTLIST Advisories %TStamp;>
<!-- A SIGMET advisory -->
<!ELEMENT SIGMET (VALID, AFFECTING?, EXTENT, BODY) >
<!ATTLIST SIGMET
class (CONVECTIVE| HOTEL| INDIA| UNIFORM| VICTOR| WHISKEY)
#REQUIRED
id NMToken #REQUIRED
%TStamp;
%BBBox-OPT;
>
<!ELEMENT AFFECTING (#PCDATA)>
<!ELEMENT EXTENT (#PCDATA)>
<!ATTLIST EXTENT
Shape (AREA| LINE| POINT) #REQUIRED
%LatLons;
>
<!ELEMENT BODY (#PCDATA)>
<!-- A collection of weather forecasts -->
<!ELEMENT Forecasts ( TAF )* >
<!ATTLIST Forecasts %TStamp;>
<!-- A Terminal Aerodrome Forecast -->
<!ELEMENT TAF ( VALID, PERIOD+ ) >
<!ATTLIST TAF
%TStamp; %LatLon; %BId; %SName;
>
<!ELEMENT PERIOD ( PREVAILING, VAR* )>
<!ATTLIST PERIOD
%TRange;
Title NMToken #IMPLIED
>
<!ELEMENT PREVAILING (#PCDATA)>
<!ELEMENT VAR (#PCDATA)>
<!ATTLIST VAR
%TRange;
Title CDATA #REQUIRED
>
<!-- Rawinsonde and Pibal Observation reports -->

```

```

<!ELEMENT UAR ( UAPART+, UAID*, UACODE*, UALEVELS ) >
<!ATTLIST UAR
%TStamp; %LatLon; %BId; %SName; %Elev;
>
<!ELEMENT UAPART (#PCDATA)>
<!ATTLIST UAPART
id NMTOKEN #REQUIRED
>
<!ENTITY % UARef "Ref NMTOKEN #REQUIRED">
<!ELEMENT UAID (#PCDATA)>
<!ATTLIST UAID %UARef; >
<!ELEMENT UACODE (#PCDATA)>
<!ATTLIST UACODE %UARef; >
<!ELEMENT UALEVELS (UALEVEL)*>
<!ELEMENT UALEVEL (#PCDATA)>
<!ATTLIST UALEVEL
%UARef;
P NMTOKEN #REQUIRED
H NMTOKEN #IMPLIED
T NMTOKEN #IMPLIED
DP NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
>
<!-- Bathythermal, Salinity and Ocean Currents Observations
-->
<!ELEMENT BTSC ( BTID, BTCODE?, BTLEVELS ) >
<!ATTLIST BTSC
%TStamp; %LatLon; %BId; %SName;
Title (JJYY | KKXX | NNXX) #REQUIRED
Depth NMTOKEN #IMPLIED
>
<!ELEMENT BTID (#PCDATA)>
<!ATTLIST BTID
DZ (7|8) #IMPLIED
Rec NMTOKEN #IMPLIED
WS (0|1|2|3) #IMPLIED
Curr-s (2|3|4) #IMPLIED
Curr-d NMTOKEN #IMPLIED
AV-T (0|1|2|3) #IMPLIED
AV-Sal (0|1|2|3) #IMPLIED
AV-Curr (0|1|2|3) #IMPLIED
Sal (1|2|3) #IMPLIED
>
<!ELEMENT BTCODE (#PCDATA)>
<!ELEMENT BTLEVELS (BTAIR?, (BTLEVEL)*)>
<!ELEMENT BTAIR (#PCDATA)>
<!ATTLIST BTAIR
T NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
>
<!ELEMENT BTLEVEL (#PCDATA)>

```

```

<!ATTLIST BTLEVEL
D NMTOKEN #REQUIRED
T NMTOKEN #IMPLIED
S NMTOKEN #IMPLIED
Curr CDATA #IMPLIED
>
<!-- Surface Synoptic Reports from land and sea stations -->
<!ELEMENT SYN ( SYID, SYCODE?, SYG?, SYSEA? ) >
<!ATTLIST SYN
%TStamp; %LatLon; %BId; %SName; %Elev;
Title (AAXX | BBXX | ZZZY) #REQUIRED
SType (AUTO | MANN) "MANN"
>
<!ELEMENT SYID (#PCDATA)>
<!ATTLIST SYID
WS (0|1|3|4) #IMPLIED
>
<!ELEMENT SYCODE (#PCDATA)>
<!ELEMENT SYG (#PCDATA)>
<!ATTLIST SYG
T NMTOKEN #IMPLIED
TD NMTOKEN #IMPLIED
Hum NMTOKEN #IMPLIED
Tmm CDATA #IMPLIED
P NMTOKEN #IMPLIED
P0 NMTOKEN #IMPLIED
Pd NMTOKENS #IMPLIED
Vis NMTOKEN #IMPLIED
Ceiling NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
WX CDATA #IMPLIED
Prec CDATA #IMPLIED
Clouds CDATA #IMPLIED
>
<!ELEMENT SYSEA (#PCDATA)>
<!ATTLIST SYSEA
T NMTOKEN #IMPLIED
Wave CDATA #IMPLIED
SDir CDATA #IMPLIED
>
<!-- Plain-text WMO Meteorological messages -->
<!ELEMENT Messages ( MSG )* >
<!ATTLIST Messages %TStamp;>
<!ELEMENT MSG ANY >
<!ATTLIST MSG
id NMTOKEN #REQUIRED
Type NMTOKEN #IMPLIED
%TStamp;
%SName;
%BBBox-OPT;
BBB CDATA #IMPLIED

```

Descr CDATE #IMPLIED

>

<!-- j> -->

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX F. SYSTEMS REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS SPECIFICATION
FOR AN
ARCHITECTURAL FRAMEWORK
OF
DOD COTS/LEGACY SYSTEM

1. SCOPE

1.1 INTRODUCTION

The trend towards using Commercial Off-The-Shelf (COTS) software within Department of Defense (DoD) has become the accepted way to build systems. Twenty years ago, almost all DoD software-intensive systems were built by awarding large multimillion-dollar contracts to defense contractors to build these systems from scratch. In the 90's, with a constantly dwindling budget, the focus has shifted to building software-intensive systems by integrating COTS software components.

Building software systems from COTS components is quite different. The black box nature of the COTS software components along with the uncontrollable evolution process requires a different architectural approach in developing systems with COTS.

1.2 PURPOSE

The purpose of this requirements specification is to analyze and document the requirements in developing an architectural framework for COTS/Legacy systems within the DoD. To focus the requirements of the architectural framework, a DoD Meteorological and Oceanographic (METOC) system, the Naval Integrated Tactical Environmental System I

(NITES I), which is very representative of today's DoD COTS/Legacy systems, will be used.

1.3 BACKGROUND

The NITES I project is a Space and Naval Warfare (SPAWAR) sponsored project within DoD. Like most other projects within DoD, the NITES I project is being developed in an environment that emphasizes the use of personal computers and COTS components.

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps forecasters. The purpose of NITES I is to provide the METOC community (Users) with the tools necessary to support the warfighter (Customers).

The NITES I is the primary METOC data fusion platform and principal METOC analysis workstation, intended to be operated on both a classified and unclassified network environment by METOC personnel. This system receives, processes, stores and disseminates METOC data and provides analysis tools to render products for application to military and tactical operations. NITES I data and information/products are stored in a unified METOC database on the C4ISR network and available to local and remote planners and warfighters.

1.4 REFERENCES

Performance Specification (PS) for the Tactical
Environmental Support System / Next Century TESS(NC)
(AN/UMK-3) (NITES version I and II)

Security Guidelines for Space and Naval Warfare Systems
Command (SPAWAR) Program Software Developers (DRAFT),
October 1999.

Horizontal Integration: Windows NT Developer's Guidelines
(DRAFT), Version 0.1.

2. GENERAL DESCRIPTION

2.1 ARCHITECTURE GOALS

Integration

COTS/GOTS/legacy components are usually created as standalone products. When these components are targeted for integration into a system, the architecture shall provide seamless integration of these COTS/GOTS/legacy components. The architecture shall support middleware approaches to bind data, information and COTS/GOTS/legacy components.

Because evolution and upgrade of COTS/GOTS components are outside the control of the system integrators, the architecture of the COTS/GOTS/legacy system shall have an adaptable component configuration to reduce the effort of testing and reintegration when upgrades or new COTS/GOTS packages are introduced to the system.

INTEROPERABILITY

COTS/GOTS and legacy systems reside on multiple platforms. This architecture shall address distributed, heterogeneous systems consisting of both UNIX and PC-based platforms.

In order to achieve and maintain information superiority on the battlefield, the architectural framework for DoD

COTS/GOTS/legacy systems shall have the capability to share, receive and transmit on heterogeneous networks and hardware devices.

The exchange of data between two systems shall be in such a way that interpretation of the data is precisely the same. The data displayed on two different systems shall remain consistent. The architectural framework shall include standard application program interfaces (APIs). APIs specify a complete interface between the application software and the platform across which all services are provided. A rigorous definition of the interface results in application portability provided the platform supports the API as specified, and the application uses the specified API. The API definitions shall include the syntax and semantics of the programmatic interface as well as the necessary protocol and data structure definitions.

ADOPTED FRAMEWORK TECHNOLOGY

Java/C++, web technologies, open systems, application program interfaces, common operating environment, object and component technology, commercial products and standards are all important to the COTS/GOTS/legacy system architecture.

The COTS/GOTS/legacy system shall adopt the Interface Definition Language (IDL) as the language for expressing the syntax of the framework services.

The COTS/GOTS/legacy system architecture shall be expressed as UML class and package diagrams, with detailed component descriptions using IDL with English narrative to provide semantics.

SECURITY

DoD tactical systems are normally classified to some security level. In building this architectural framework, the architecture shall address the DoD Trusted Computer System Evaluation Criteria (TCSEC) to at least the C2 security level.

The architecture shall include discretionary access control (DAC).

Only single level classification systems shall be supported in this architecture (i.e. no multi-level security (MLS)).

Assembled components shall not require modification to add security services.

The security mechanisms shall be protected from unauthorized access.

The following security services shall be available to the component assembler:

1. Single login for users

The single login for users means the user needs to identify himself once per session. It is the responsibility of the security services to protect and distribute the authentication information of a user.

2. Mutual authentication

Mutual authentication ensures proper identification of the user to the system and the system to the user.

3. Auditing

Auditing means significant security events are recorded for later analysis. Significant security events shall include logon and logoff, security policy changes, user and group management, and access to specified objects.

4. Secure key distribution

Key distribution provides a secure transport mechanism for encryption keys.

5. Role based Access Control

Role based access control assigns roles to users and privileges to roles, thereby simplifying access control if the number of roles is less than the number of users.

6. Data confidentiality

Data confidentiality means data is disclosed according to a policy.

7. Data integrity

Data integrity means the recipient gets the intended data.

8. Non-repudiation and authenticity

Non-repudiation means the sender of a message can not later deny he sent the message.

NETWORK SECURITY

The trend in DoD is for networked systems vice standalone monolithic systems and because most systems have some level of classification, this architecture shall address network security.

The architectural framework shall support a secure network.

The architectural framework shall support the network security mechanisms specific to the target architecture, including firewalls, routers, encryption, and proxy services.

NETWORK COMMUNICATIONS

The architectural framework shall support different network protocols (i.e. TCP/IP) and topologies dependent on the target architecture.

The application layer shall be able to execute a variety of data management commands without having knowledge of the data location, database, file type, operating system, network protocol, or platform location.

DEVELOPMENT LANGUAGE

The architectural framework shall support any development language that is supported by the legacy system as well as any development language that supports platform independence for newly developed code in the target architecture.

2.2 ASSUMPTIONS AND DEPENDENCIES

Assumption 1: Legacy systems are monolithic and not modifiable.

Assumption 2: Legacy systems have some existing mechanism for interaction.

Assumption 3: There are varying degrees of COTS. To be considered COTS, the component cannot be modified.

Assumption 4: Reliability, performance, safety and security must be weighed in the target architecture.

Assumption 5: Multilevel security systems are beyond the scope of this effort.

3. TARGET ARCHITECTURE FUNCTIONS -

DATABASE

COTS software applications which handle data tend to have their own mechanism and structure for the storage of the data internal to the COTS application. When the target architecture includes a master database to store its data, the architectural framework shall support the target architecture's central storage of data. The architecture shall support remote access to the database.

SECURITY

The target architecture shall support Discretionary Access Control (DAC).

Access to information controlled by an application shall be based on an access control list (ACL) of a parameter that can be used to distinguish between authorized and non-authorized entities. Entities include users, devices, and other applications.

The target architecture shall support non-repudiation.

- a. The data recipient shall be assured of the originator's identify.
- b. The data originator shall be provided with proof of delivery.

- c. The algorithm used to digitally sign data entries and receipts shall be either the Digital Signature Standard (DSS) FIPS 186 or RSA (1024 bit).
- d. The original transmitted data signed by the sender and the requested receipt signed by the recipient shall be time-stamped by a trusted third party.

GRAPHICAL USER INTERFACE (GUI)

The target architecture shall include a GUI style guide. If a GUI style guide does not exist for the target architecture, UNIX platforms shall adhere to the MOTIF standard and X-Windows standard, and PC platforms shall adhere to the Windows NT standard.

EXTERNAL SYSTEM INTERFACES

Because the target architecture exists in a network environment where it shares data with other external systems, the external system interfaces where information is exchanged shall be well defined to support interoperability.

MIDDLEWARE TECHNOLOGY

The COTS/GOTS/legacy architecture shall support new component integration technologies (i.e. COM/DCOM) to broker between components that by themselves normally do not communicate to form an integrated system.

The target architecture shall support wrappers to enable COTS/GOTS applications to interface with each other. The wrappers shall support the METOC data (listed in Table 6 of reference 1) and its various formats within NITES. The architecture shall ensure when an application updates a set of data, the update is consistently made throughout the rest of the database.

4. ARCHITECTURE ATTRIBUTES

4.1 PERFORMANCE REQUIREMENTS

The performance requirements for the target system are contained in Table 6B of the NITES Performance Specification. In addition to those performance requirements, the following requirements shall also be addressed in the target architecture.

The architecture shall optimize the database access over a network.

The architecture shall allow concurrent access of the database to multiple users.

The component technology shall not degrade the system performance by more than 10% of the target system's current performance requirements. Refer to Table 6B of the NITES Performance Specification.

4.2 RELIABILITY REQUIREMENTS

The target architecture shall use standard fault-tolerant technologies (i.e. Replication to maintain the reliability and availability requirements of DoD systems,) While the data traverses throughout various applications, to different platforms, through the network and to/from

database, it must remain consistent and not suffer any degradation.

4.3 DESIGN CONSTRAINTS

Because many existing legacy systems reside on UNIX platforms and the DoD has made a commitment to move towards a PC architecture, the architectural framework shall support both UNIX and PC platforms with the goal of moving towards a pure PC architecture. It is not required that all COTS/GOTS/legacy system components be executable on both platforms but the data must be able to be shared by components on different platforms.

Newly developed DoD systems must use COTS products to the greatest extent possible.

As most COTS/GOTS applications are designed to be standalone, these applications will usually have their own way of retrieving and storing data. When these applications are integrated into a system, the internals of the application of how it retrieves and stores data will not be modified.

There are varying degrees of COTS products. Depending on whether the COTS product is an opaque or a black box will drive the wrapper design and implementation.

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX G. SYSTEM DESIGN SPECIFICATION

1. SYSTEM ARCHITECTURE

1.1 SYSTEM ARCHITECTURE DIAGRAM

The Naval Integrated Tactical Environmental System (NITES) software runs in a distributed, heterogeneous environment on standard commercial-off-the-shelf (COTS) personal computers (PCs) and TAC-4 UNIX computers.

The NITES architecture consists of a central database residing on a UNIX computer, which is shared amongst the various NITES components (most of which reside on PCs with the exception of the tactical applications which reside on a TAC-4 UNIX computer) as depicted in figure 1. In this topology, there is no direct interaction between the components. All interactions are through the central database. This topology allows ease of integration of COTS components as it minimizes the integration effort since each component only has one interconnection.

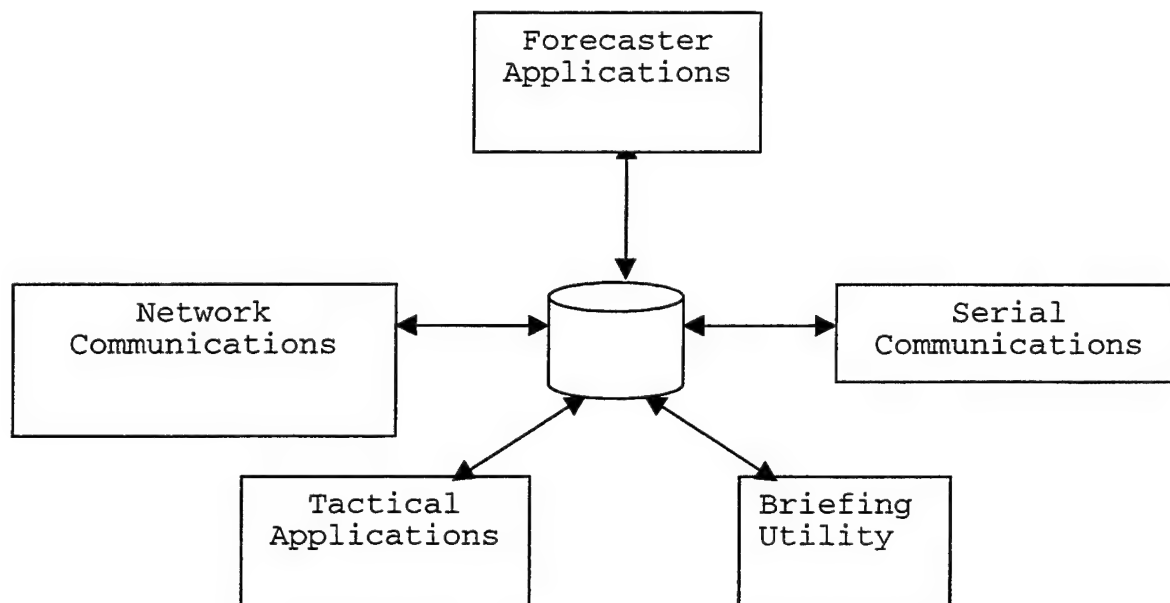


Figure 1 - NITES Architecture Diagram

Forecaster applications (COTS/GOTS) - Manipulate METOC data to easily plot, analyze, display on a common geographical reference.

Serial Communications (Legacy code) - Handles the ingest and dissemination of METOC data through existing legacy communication channels.

Briefing (COTS) - Briefing utility used to brief tactical commanders, flight operators the environmental conditions that they will be operating in.

Tactical applications (Legacy code and newly developed code) - Tactical applications take in METOC data to predict the affects of the environmental conditions on the environment, tactical equipment, etc.

Database (GOTS) - The database is the central repository for all METOC data.

Network communications (GOTS) - Handles the ingest and dissemination of METOC data through SIPRNET.

The deployment diagram, as depicted in figure 2, consists of a NITES Server, a NITES Database Server, and NITES workstations with a communications package, an applications package, a database package, a system controller package, a security package and a briefier package residing on multiple hardware platforms.

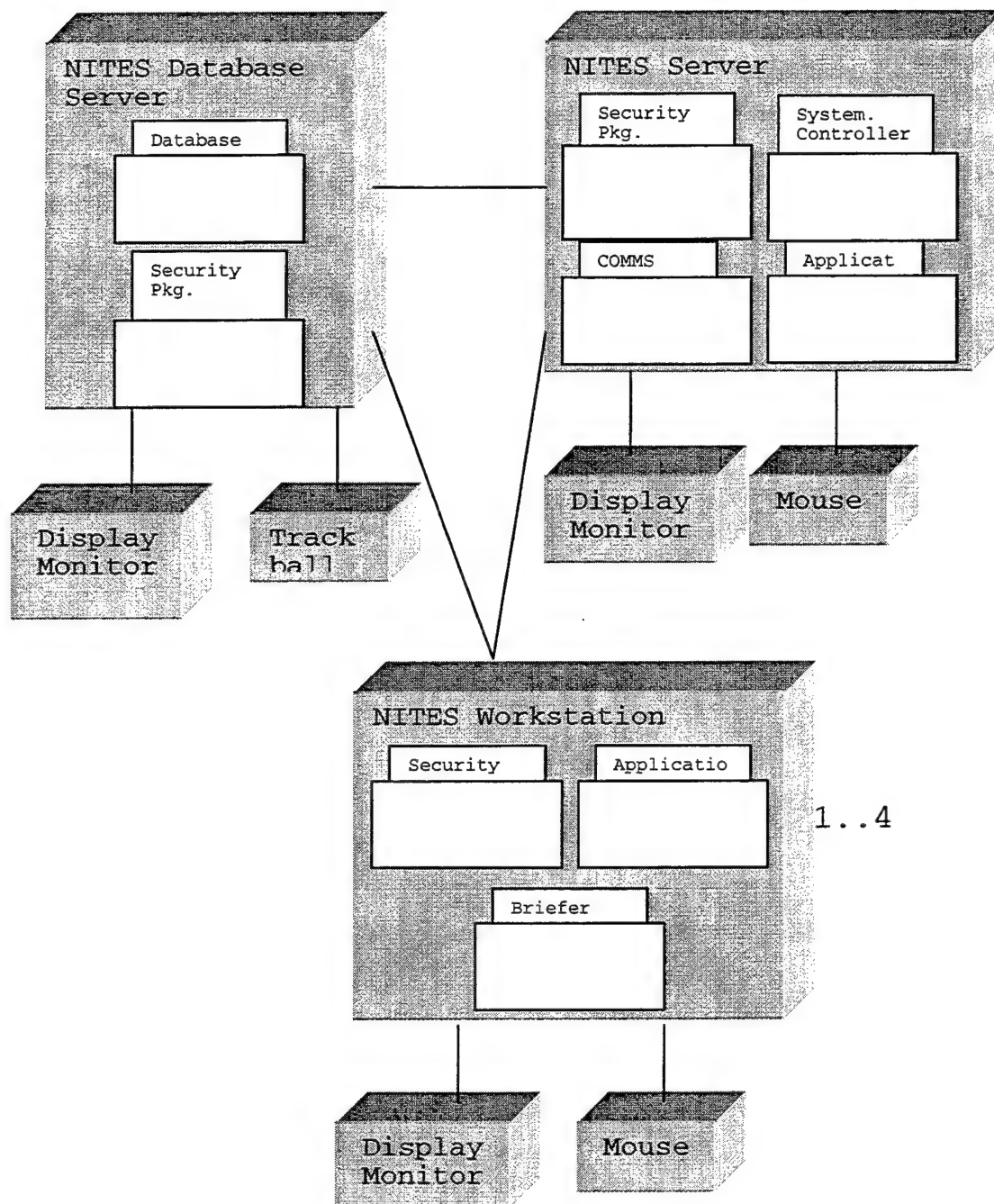


Figure 2 - Deployment Diagram

In the NITES architecture, all interactions are through the NITES database. However, in the initial delivery of the NITES software, this architecture was violated since none of the COTS applications were able to communicate with the NITES database to retrieve and/or store data and products.

A prototype of a portion of the NITES system will be developed to demonstrate the NITES architecture where a COTS application can communicate with the NITES database to retrieve and store data and products. A system controller package and the security package are newly developed for the NITES. The COTS applications packages and the briefier package will be modified to use wrapper and glue technology to enable it to communicate with the database package. These packages will be designed and developed to move the system in the direction of conforming to the existing architecture.

This prototype will use an object request broker (ORB) to marshal events/notifications in a distributed environment. Because this prototype is being developed under the Windows NT environment, and DCOM is freely available with Windows NT, we have chosen to use DCOM as our ORB.

DCOM components can communicate three ways: within the same process, out of process and between network nodes. The component internals do not need to be changed regardless of

the deployment decision. The DCOMCNFG and dynamic link library (DLL) packaging are used to implement the deployment decision.

Deployment flexibility affords alternative performance solutions in a distributed network environment. For example, the Monitor component could be deployed on a different network node than the Controller component to reduce CPU load. This solution assumes the sampling rate is higher than the notification rate.

1.2 INTER-TASK COMMUNICATION

The tasks on the NITES will be implemented to run asynchronously. Communications are broken down between the following tasks:

- Monitor/Controller
- Controller/Glue Component
- CBWrapper/Glue Component
- CBWrapper/Controller

The Application Wrapper is responsible for making the object available to a COTS viewer application.

MONITOR/CONTROLLER

Slides for the briefing package are generated by the operator using an external COTS/GOTS application. As each of these slides is generated, it is saved to a directory by

the COTS/GOTS application. The system monitor polls the directory and when a file is found, notifies the controller.

CONTROLLER/GLUE COMPONENT

When the controller receives notification from the monitor that a new file exists, the controller will create an instance of the glue component.

CBWRAPPER/CONTROLLER

CBWrapper registers interest in new products with the controller.

When the controller is notified by the glue component that a file is successfully stored in the database, it will broadcast the information to all the wrappers running on client workstations. It is the responsibility of the CBWrapper to ignore image types not appropriate for the current brief. This assumes there is at least one wrapper running.

CBWRAPPER/GLUE COMPONENT

The CBWrapper requests an image product from the glue code, which will use the existing database APIs to connect to the database, retrieves the product and returns it to the CBWrapper. The request mechanism is used to initialize and update the brief.

2. SUBSYSTEM DESCRIPTION

The object diagram and sequence diagram depicts objects required to design the update of a briefing package and the scenario of updating a briefing package in figures 3 and 4 respectively.

MONITOR

The Monitor component is responsible for detecting the presence of a new object.

CONTROLLER

The controller component is responsible for coordinating multiple concurrent asynchronous activities. The controller runs on the application server. It serves two functions within the system, handling notifications from the monitor and the glue component.

GLUE COMPONENT

The glue component is responsible for storing and retrieving objects from an ODBC compliant relational database.

CBWRAPPER

Wrappers are software code developed to add, modify, and hide functionality from COTS, GOTS or legacy software

components to align them with the overall system requirements and architecture. In the design, wrapper and glue code technology is being implemented to enable the COTS applications to adhere to the existing NITES architecture.

The briefing package consists of Microsoft PowerPoint, a COTS application package. The PowerPoint application contains APIs, which can be used by CBWrapper to create the added functionality of automatically creating and updating the briefing package in the background.

The PPT APIs used for the wrapper interface include:

- Presentations.Add
- Slides.Add
- SlideShowTransition
- SlideShowSetting
- Shapes.AddPicture
- Shapes.PictureFormat

INITIALIZATION GUI

The Initialization GUI is used to initialize each component with the number of images, starting from the most current; the image type; the display duration of each image in seconds; and the height and width of the display area. Default values are 24 images, 0 second duration, and display area equal to the workstation's screen size.

CONFIGURATION GUI

The Configuration GUI defines the set of image types available for the brief. Associated with each image type is the working directory containing the current set of brief images and a web server virtual directory corresponding to the working directory. The CBWrapper uses the configuration file to initialize the image type options available to the briefer. The monitor uses the configuration file to build a list of directories to poll.

The Configuration GUI is not restricted to the image types settings. It can be used for defining various sets of key values. For instance, we can use this Configuration GUI to define the key set values for network configuration, or application's initial default settings. This provides the extensibility for future development of applications.

NAMING CONVENTION

The filename associated with each image type consists of the fields represented the created date and time, the file format (i.e., gif, jpeg, etc.), and other information for a particular image (i.e., the channel, the location, etc.)

The filename begins with the date and time, followed by other information. For instance, a file named "20000523.1331.gms5.IR.MODEL_OVERLAY.500HT.NOGAPS" indicates that the file was created on May 23, 2000, at 13:31. The

CBWrapper uses the date and time embedded in the filename for updating the continuous brief.

The other information of the filename is used by the Glue component for storing and retrieving images to and from the database.

THIN CLIENT TECHNOLOGY

CBWrapper is implemented using modern thin client technology. When a user opens a HTTP page from a browser, the CBWrapper is then automatically downloaded and installed on the client machine. Once the CBWrapper is up and running, all images needed for creating the brief are dynamically downloaded from the server using the OpenURL method. OpenURL uses the current open HTTP connection to transfer image files. The continuous brief is created on the client machine using the PowerPoint APIs. The PowerPoint is used to display the brief.

PUSH TECHNOLOGY

The advantage of using this technique is that the client needs not to poll the server periodically for new data. The server notifies its clients (CBWrapper) when new data (images) arrive. The CBWrapper receives the notification and compares the image type with the type being showed. If the image types match, the CBWrapper downloads a new set of images from the server and updates the brief.

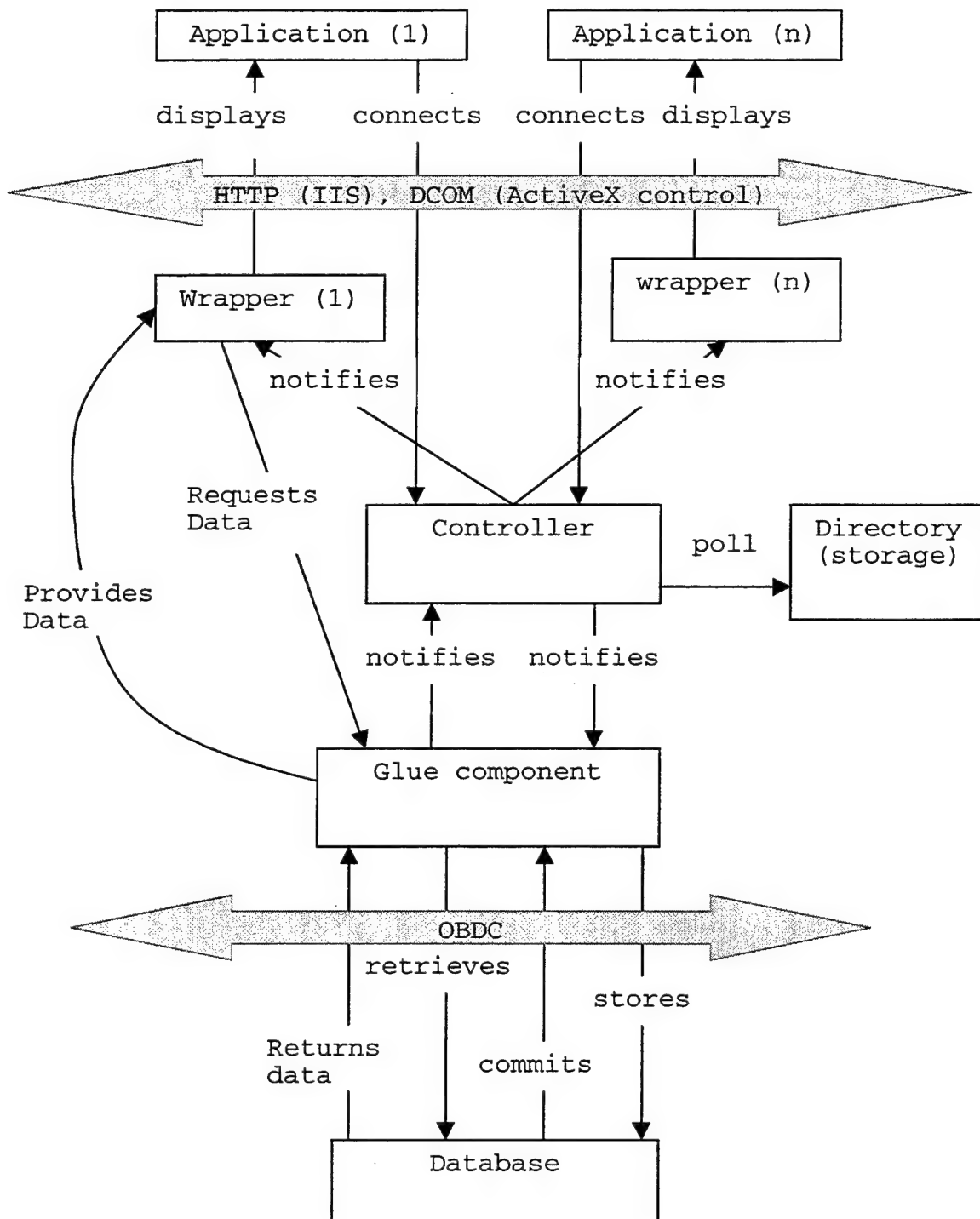


Figure 3 - Wrapper & Glue Code Object Diagram

OMF

Sharing different formatted data requires a common representation of data to interpret, send, and receive any data, any format, anywhere. Within NITES, meteorological and oceanographic observations, and certain types of bulletins (SIGMETS, JOTS warnings, and Tropical Cyclone Warnings, for example) are received and transmitted in an Extensible Markup Language (XML)-based format called Weather Observation Markup Format (OMF). OMF preserves the original text of each observation or bulletin, and also includes information decoded from the observation/bulletin and other metadata concerning the message.

OMF solves the data interoperability problem by providing self-describing tags along with the data so that the receiving applications can consistently interpret the data correctly. These self-describing tags are detailed in the Document Type Definition (DTD). When drafting the NITES data into OMF, three things must be agreed on: which tags will be allowed, how tagged elements may nest within one another and how they should be processed. The first two, the language's vocabulary and structure, are codified in the DTD.

OMF is an application of XML, and by its virtue, an application of SGML. SGML is used extensively within DoD for documenting of various types of information (military

standards, procurement materials, service manuals). OMF brings weather observations into the same fold. Thus, the design goals of OMF are:

- Mark up (annotate) raw observation reports with additional description and derived, computed quantities.
- The raw report data must not be modified in any way, and should be conveniently extractable (by simply stripping all the tags away).
- OMF must be concise. While providing useful annotations to a client, OMF markup should not impose undue overhead on communication channels.
- It should be possible to extend the markup with additional annotations, without affecting applications that do not use this information.

The OMF contains the following elements:

- **Reports** - defines a group of weather observation reports
- **METAR** for a single METAR report
- **SPECI** for a single SPECI report
- **UAR** for a combined Rawinsonde and Pibal Observation report
- **BTSC** for ocean profile data (temperature, salinity, current)

- **SYN** for a surface synoptic report from a land or sea station
- **Advisories** - defines a collection of weather hazard warnings
- **SIGMET** - SIGnificant METeorological Information
- **Forecasts** - defines a set of weather forecasts
- **TAF** - Terminal Aerodrome Forecasts
- **Messages** - defines a set of plain-text bulletins.

The following sections define the major elements along with the minor elements that are relevant to them. In each section, XML DTD declarations are provided for precise definition of elements and attributes. The collection of XML DTD declarations found in this specification can be arbitrarily extended to add new elements and attributes for new enhancements. Some of the element attributes are common. For compactness, they are defined in the following table.

Table 1-1. Basic Attributes of an Observation in OMF

Attribute	Brief Description	Format	Description
TStamp	Time Stamp	unsigned integer	UTC time in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. This is the value returned by a POSIX function time(2). Example: Tstamp='937507702'
TRange	Time Interval	a string of form "aaa, bbb", where aaa and bbb are unsigned integer numbers specifying the beginning and the end timestamps of the interval.	Timestamps are in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. These are the values returned by a POSIX function time(2). Example: Trange='937832400, 937915200'

LatLon	Specificati on of a Point on the globe	A string of a form "aaa.bbb, ccc.ddd", where aaa.bbb and ccc.ddd are signed floating point numbers	<p>The latitude and. longitude, respectively, of a point on the globe, in whole and fractional degrees. The numbers are positive for Northern latitudes and Eastern longitudes, and negative for Southern latitudes and Western longitudes.</p> <p>The range of the numbers is [-90.0, 90.0] for latitudes, [-180.0, 180.0] for longitudes.</p> <p>Example: LatLon='32.433, - 99.850'</p>
LatLons	Specificati on of a Sequence of Points on the Globe	a string of a form "lat1, lon1, lat2, lon2, latn, lonn" where each pair (lat1, lon1, etc.) are signed floating point numbers	<p>A sequence of pairs of numbers, each pair giving the latitude and longitude of a single point in the sequence, in whole and fractional degrees.</p> <p>See the LatLon attribute above for more details.</p> <p>Example: LatLons='38.420, - 111.125, 36.286, - 111.492, 36.307, - 112.630, 37.700, - 113.223, 38.420, - 111.125'</p>

Table 1-1. Basic Attributes of an Observation in OMF

Attribute	Brief Description	Format	Description
BBox	Bounding box, which tells the latitudinal and the longitudinal spans of an area of the globe	A string of a form "lat-N, lon-W, lat-S, lon-E", where the lats and lons are signed floating-point numbers, in degrees	<p>Specification of the bounding box for an area of interest. Here lat-N is the latitude of the Northern-most point of the area, lat-S is the latitude of the Southern-most point, lon-W is the longitude of the Western-most point of the area, and lon-E is the Eastern-most longitude.</p> <p>It is required that lat-N \geq lat-S. The left-lon (lon-W) may however be greater than the right-lon (lon-E). For example, a range of longitudes [-170,170] specifies the entire world but Indonesia. On the other end, the range [170, -170] includes Indonesia only. By the same token, [-10,10] pertains to a 21-degree longitude strip along the Greenwich meridian, while [10,-10] specifies the whole globe except for that strip.</p> <p>Example: Bbox='60.0, -120.0, 20.0, -100.0'</p>
BId	Station identificat	Unsigned integer	WMO Block Station ID, or other

	ion group		identifier for buoy or ship
SName	Call sign and full name of an observing station	A string of the form "ccccc, name", where ccccc are the call letters of the station (ICAO station id: 4 or 5 upper-case letters, may be omitted), name is an arbitrary string describing the station	The observing stations ICAO, aircraft, or ship call sign, plus a plain-text station name (e.g. "KMRY, Monterey CA Airport" Example: Sname='KYNL, YUMA (MCAS)'
Elev	Elevation	A non-negative integer, or omitted if unknown.	Station elevation relative to sea level, in meters. This attribute may specify a surface elevation of an observation station, or an upper-air elevation for an upper-air report. Example: Elev='16'

Table 1-2. OMF Attributes for METAR and SPECI Reports

Attribute	Brief Description	Format	Description	Req'd ?
TStamp	Time Stamp	<-----See Table 1-1-- ----->		Yes
LatLon	Station latitude and longitude	<-----See Table 1-1-- ----->		Yes
BId	Station Identification Group	Unsigned integer	WMO Block Station ID	Yes
SName	Call sign and full name of an observing station	<-----See Table 1-1-- ----->		Yes
Elev	Station elevation	<-----See Table 1-1-- ----->		No
Vis	Visibility	a number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	No
Ceiling	Ceiling	a number of feet, omitted, or a special token "INF"	Ceiling in feet	No

Table 1-3. OMF Attributes for the SYN Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<-----See Table 1-1----->		Yes
LatLon	Station latitude and longitude	<-----See Table 1-1----->		Yes

BId	WMO Block Station Number	String	<p>For a buoy or other observation platform, this id is a combination of a WMO region number, subarea number (per WMO Code Table 0161), and the buoy type and serial number. This information is reported in Section 0 of a synoptic report.</p> <p>If Section 0 contains a call sign rather than a numerical id (as typical with FM 13 SHIP reports), the BId attribute is computed as $\text{itoe}(1000009 + \text{hc}) \% 2^{30}$, where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the Bid attribute a unique numeric identifier for the station.</p>	Yes
SName	Call sign and full name of an observing station	<-----See Table 1-1----->		Yes
Elev	Station elevation	<-----See Table 1-1----->		No

Title	Report title	String	Title defining type of report: AAXX (FM-12), BBXX (FM-13), or ZZZY (FM-18)	Yes
Stype	Station type	String	Type of station: automated (AUTO) or manned (MANN); defaults to MANN	No

Table 1-4. OMF Attributes for the SYG Element

Attribute	Brief Description	Format	Description	Req'd?
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius	No
TD	Dew point Temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius	No
Hum	Relative humidity	non-negative number	Relative humidity in per cent	No
Tmm	Extreme temperatures over the last 24 hours	a string of a form "mmmm, MMMM" or omitted	Minimum and maximum temperatures (degrees Celsius) over the last 24 hours	No
P	Station pressure	positive number	Atmospheric pressure at station level, in hectoPascals	No
P0	Sea level pressure	positive number	Atmospheric pressure at station, reduced to sea level, in hPa	No
Pd	Pressure Tendency	String of form "dddd", or omitted	Pressure tendency during the 3 hours preceding the observation	No
Vis	Visibility Number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	Horizontal visibility in meters	No
Ceiling	Ceiling	Number of feet, omitted, or a special token "INF"	Ceiling in feet	No
Wind	Wind speed and direction	String of form "nnn, mm" or omitted	nnn is a true direction from which the wind is blowing, in degrees, or VAR if	No

			<p>" the wind is variable, or all directions or unknown or waves confused, direction indeterminate."</p> <p>This is an integer number within [0,360), with 0 meaning the wind is blowing from true North, 270 stands for the wind blowing from due West. Normally this number has a precision of 10 degrees.</p> <p>mm is the wind speed in meters per second.</p>	
--	--	--	--	--

Table 1-4. OMF Attributes for the SYG Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
Wx	Past and present Weather conditions and phenomena	String of four digits, "NOSIG", or omitted	See WMO-306, Code tables 4677 and 4561 for the meaning of the four digits. This attribute is coded as "NOSIG" if there is no significant phenomenon to report. The attribute is omitted if not observed or data is not available (see ix indicator, Code table 1860).	No

Prec	Precipitation amount	String of form "nnn, hh" or "" or omitted	nnn is the amount of precipitation which has fallen during the period preceding the time of observation. The precipitation amount is a non-negative decimal number, in mm. hh is the duration of the period in which the reported precipitation occurred, in whole hours. This attribute is encoded as "" if no precipitation was observed. The attribute is omitted if unknown or not available (see <small>ir</small> indicator, Code table 1819). Sea stations typically never report precipitation.	No
Clouds	Amounts and types of cloud cover	String of five symbols "tplmh" or omitted	The first digit is the total cloud cover in octas (Code table 2700). The second digit is the cloud cover of the lowest clouds, in octas. The other three symbols are types of low, middle, and high clouds, resp. See WMO-306 Code tables for more details.	No
T	Sea surface temperature	Positive, zero, or negative number	Sea surface temperature in degrees Celsius	No
Wave	Sea wave period and height	String of form "pp, hh" or	pp is the period of wind waves	No

		omitted	in seconds. hh is the height of wind waves, in meters. If a report carries both estimated and measured wind wave data, the instrumented information is preferred.	
--	--	---------	---	--

Table 1-4. OMF Attributes for the SYG Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
SDir	Ship's course and speed	String of form "nnn, mm" or omitted.	nnn is a true direction of resultant displacement of the ship during the three hours preceding the time of observation. The number is in degrees, or VAR if "variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the ship has moved towards the true North; 270 means the ship has moved to the West. Normally this number has a precision of 45 degrees.	No

			mm is the average speed made good during the three hours preceding the time of observation, in meters per second.	
--	--	--	---	--

Table 1-6. OMF Attributes for the UALEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
Ref	Reference to sounding Part	String - "TTAA", "TTBB", etc.	Reference to the part of the sounding from which the level data were derived	Yes
P	Pressure	positive number	Atmospheric pressure at sounding level, in hectoPascals	Yes
H	Geopotential height	Non-negative number of geopotential meters, or 'SURF' for surface, 'TROP' for tropopause, 'MAXW' for level of maximum winds, 'MAXWTOP' for maximum wind level at the top of the sounding, or omitted	Geopotential height of the reported level, or a special height indicator	No
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius at the reported level	No
DP	Dew point temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius at the reported level	No
Wind	Wind speed and direction	String of form "nnn, mm" or "nnn, mm bbb" or "nnn, mm ,aaa" or "nnn, mm	nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or	No

		bbb, aaa" or omitted	<p>unknown or waves confused, direction indeterminate."</p> <p>This is an integer number within [0,360), with 0 meaning the wind is blowing from true North, 270 stands for the wind blowing from due West. Normally this number has a precision of 10 degrees.</p> <p>mm is the wind speed in meters per second.</p> <p>If specified, bbb stands for the absolute value of the vector difference between the wind at a given level, and the wind 1 km below that level, in meters per second. The number aaa if given is the absolute value of the vector difference between the wind at a given level, and the wind 1 km above that level, in meters per second.</p>	
--	--	----------------------	--	--

Table 1-7. OMF Attributes for the BTSC Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<----- See Table 1-1 -----> ----		Yes
LatLon	Latitude and Longitude of observation	<----- See Table 1-1 -----> ----		Yes
BId	Station identifier group	positive integer	For a buoy or other observation platform, this ID is a combination of a WMO region number, subarea number (per WMO-306 Code Table 0161), and the buoy type and serial number. This information is reported in Section 4 of a BTSC report. If Section 4 contains a call sign rather than a numerical id, the BId attribute is computed as $\text{itoe}(1000009 + \text{hc})$, where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the Bid attribute a unique numeric identifier for the station.	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes
Title	Report type	string	"JJYY" - FM 63 X Ext. BATHY report	Yes

			"KKXX" - FM 64 IX TESAC report "NNXX" - FM 62 TRACKOB report	
Depth	Water depth	positive number	Total water depth at point of observation	No

Table 1-8. OMF Attributes for the BTID Element

Attribute	Brief Description	Format	Description	Req'd?
DZ	Indicator for digitization	"7" or "8" or omitted	Indicator for method of digitization used in the report (k_1 field). See WMO-306 Code Table 2262. Required for BATHY and TESAC reports	No
Rec	Instrument type code	5-digit code	Code for expendable bathythermograph (XBT) instrument type and fall rate (WMO-306 Code Table 1770)	No
WS	Wind speed units code	"0", "1", "2", "3", or omitted	Indicator for units of wind speed and type of instrumentation (i_u field). See WMO-306, Code Table 1853.	No
Curr-s	Method of current speed measurement	"2", "3", "4", or omitted	Indicator for the method of current measurement (k_5 field). See WMO-306 Code Table 2266.	No
Curr-d	Indicators for the method of subsurface Current measurement	3-digit numerical code	Indicators for the method of subsurface current measurement ($k_6k_4k_3$ codes). See WMO-306, Code Tables 2267, 2265, and 2264.	No
AV-T	Averaging period for sea temperature	"0", "1", "2", "3", or omitted (if no sea temperature data are reported)	Code for the averaging period for sea temperature (m_T code). See WMO-306, Code Table 2604	No
AV-SAL	Averaging period for salinity.	"0", "1", "2", "3", or omitted	Code for the averaging period for sea salinity (m_S code).	No

		(if no salinity data are reported)	See WMO-306, Code Table 2604	
AB-Curr	Averaging period for surface Current direction and speed	"0", "1", "2", "3", or omitted (if no current data are reported)	Code for the averaging period for surface current direction and speed (_{mc} code). See WMO-306, Code Table 2604	No
Sal	Method of salinity/depth measurement	"1", "2", "3", or omitted (if no salinity data are reported)	Code for the method of salinity/depth measurement (_{k2} code). See WMO-306, Code Table 2263.	No

Table 1-9. OMF Attributes for the BTAIR Element

Attribute	Brief Description	Format	Description	Req'd?
T	Air temperature	Positive, zero, or negative number, or omitted	Air temperature just above the sea surface, in degrees Celsius.	No
Wind	Wind vector	String of form "nnn,mm", or omitted	Here nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from the true North;; 270 means the wind is blowing from the West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second.	No

Table 1-10. OMF Attributes for the BTLEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
D	Depth	Non-negative number	Depth of the level in meters.	Yes
T	Water temperature	Positive, zero, or negative number, or omitted	Water temperature at the reported level.	No
S	Salinity	Positive number, or omitted	Salinity at the reported level, in parts per thousand.	No

C	Current vector String of form	"nnn,mm", or omitted	nnn is the true direction toward which the sea current is moving, in degrees, or VAR if "the current is variable, or all directions or unknown, direction indeterminate." This is an integer number within [0,360), with 0 meaning the current flows toward true North; 270 means the current is flowing toward the West. Normally this number has a precision of 10 degrees. mm is the speed of current in meters per second.	No
---	----------------------------------	----------------------------	---	----

Table 1-11. OMF Attributes for the TAF Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<----->	See Table 1-1 ----->	Yes
LatLon	Latitude and Longitude of observation	<----->	See Table 1-1 ----->	Yes
BId	Block Station ID	positive integer	WMO Block Station ID of the reporting station	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes

Table 1-12. OMF Attributes for the SIGMET Element

Attribute	Brief Description	Format	Description	Req'd?
class	SIGMET type	"CONVECTIVE",	Identifier for the type of SIGMET	Yes

		"HOTEL", "INDIA", "UNIFORM", "VICTOR", "WHISKEY"	message -	
id	Identifier for a particular advisory	String	Identifier for the advisory; value depends on the advisory class.	Yes
TStamp	Time Stamp	<----- See Table 1-1 ----- ->		Yes
BBox	Bounding box for advisory area	<----- See Table 1-1 ----- ->		Yes

Table 1-13. OMF Attributes for the EXTENT Element

Attribute	Brief Description	Format	Description	Req'd?
Shape	Type of area specificati on	"AREA", "LINE", "POINT"	Type of area shape specified	Yes
LatLons	List of latitudes and Longitudes defining the area	Positive, zero, or Negative numbers in lat/lon pairs	Control points (vertices) for a polygon/polyline representing the affected area	Yes

Table 1-14. OMF Attributes for the MSG Element

Attribute	Brief Description	Format	Description	Req'd?
id	Message identifier	A NMTOKEN, a four-to-six-character string of a form $\tau_1\tau_2\alpha_1\alpha_2ii$	Designator for the message type and subtype ($\tau_1\tau_2$), area ($\alpha_1\alpha_2$), and sequence code (ii) of the message, as described in WMO-386.	Yes
Type	Message type	2-letter string ($\tau_1\tau_2$)	Designator for the message type and subtype ($\tau_1\tau_2$) as specified in WMO-386, Tables A and B1 through B6	Yes
TStamp	Time Stamp	<-----	See Table 1-1 ----- -->	Yes
SName	Originating station name	String	String containing the identification of the station that originated the message (normally its ICAO call sign)	Yes
BBB	Annotation group	3-character string	So-called "BBB groups" from the abbreviated message line. They indicate that the message has been delayed, corrected or amended. A BBB group can also be used for segmentation. See the WMO-386 for more detail.	No
Descr	Description	String	Keywords and other information describing the message.	No
BBox	Bounding box	<-----	See Table 1-1 ----- -->	No

Table 1-15 Layer Parameter Codes

layer	Description	Example
adiabatic-cond	Adiabatic condensation level (parcel lifted from surface)	(layer adiabatic-cond)
atm-top	Level of the top of the atmosphere	(layer atm-top)
cloud-base	Cloud base level	(layer cloud-base)
cloud-top	Cloud top level	(layer cloud-top)
conv-cld-base	Level of bases of convective clouds	(layer conv-cld-base)
conv-cld-top	Level of tops of convective clouds	(layer conv-cld-top)
entire-atm	Entire atmosphere	(layer entire-atm)
entire-ocean	Entire ocean	(layer entire-ocean)
height	Height above ground (meters)	(layer height 1500)
height-between	Layer between two heights above ground in hundreds meters (followed by top and bottom level values)	(layer height-between 50 30) for layer between 5000 and 3000 meters above ground
height-between-ft	Layer between two heights above ground, in feet (followed by top and bottom level values)	(layer height-between-ft 15000 10000)
height-ft	Height above ground (feet)	(layer height-ft 50)
high-cld-base	Level of high cloud bases	(layer high-cld-base)
high-cld-top	Level of high cloud tops	(layer high-cld-top)
hybrid	Hybrid level (followed by level number)	(layer hybrid 1)
hybrid-between	Layer between two hybrid levels (followed by top and bottom level numbers)	(layer hybrid 2 1)
isobar	Level of an isobaric surface (followed by the	(layer isobar 500)

	isobar value of the surface in hectoPascals (hPa) (1000, 975, 950, 925,900,850,800,750,7 00,65 0,600,550,500,450,400 ,350,3 00,250,200, 150,100, 70, 50, 30, 20,10)	-
isobar-between	Layer between two isobaric surfaces (followed by top and bottom isobar values in kPa, separated by a space)	(layer isobar- between 50 100) for layer between 500 and 1000 hPa
isobar-between-mp	Layer between two isobaric surfaces, mixed precision (followed by pressure of top in kPa and 1100 minus pressure of bottom in hPa)	(layer isobar- between-mp 50 100) for layer between 500 and 1000 hPa

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
isobar-between-xp	Layer between two isobaric surfaces, extra precision (followed by top and bottom isobar values expressed as 1100 hPa-isobar level, separated by a space)	(layer isobar-between 600 100) for layer between 500 and 1000 hPa
isotherm-0	Level of the zero-degree (Celsius) isotherm (or freezing level)	(layer isotherm-0)
land-depth	Depth below land surface in centimeters	(layer land-depth 5.0)
land-depth-between	Layer between two depths in ground (followed by the depth of the top of the layer and the depth of the bottom of the layer centimeters)	(layer land-depth-between 0 30) for layer from ground surface to 30 cm depth
land-height-cm	Height level above ground (high precision) (followed by height in centimeters)	(layer land-height-cm 50)
land-isobar	Pressure above ground level in hPa	(layer land-isobar 500)
land-isobar-between	Layer between two isobars above levels (followed by top and bottom isobaric levels in hPa)	(layer land-isobar-between 500 1000)
low-cld-base	Level of low cloud bases	(layer low-cld-base)
low-cld-top	Level of low cloud tops	(layer low-cld-top)
max-wind	Level of maximum wind	(layer max-wind)
mid-cld-base	Level of middle cloud bases	(layer mid-cld-base)
mid-cld-top	Level of middle cloud tops	(layer mid-cld-top)
msl	Mean sea level	(layer msl)
msl-height	Height above mean sea	(layer msl-height

	level (in meters)	50)
msl-height-between	Layer between two heights above mean sea level in hundreds of meters (followed by top and bottom height values)	(layer msl-height-between 10 5) for layer between 1000 and 500 meters above ground
msl-height-ft	Height above mean sea level (in feet)	(layer msl-height-ft 5000)
sea-bottom	Bottom of the ocean	(layer sea-bottom)
sea-depth	Depth below the sea surface (meters)	(layer sea-depth 50)
sigma	Sigma level in 1/10000	(layer sigma 9950) for sigma level .995
sigma-between	Layer between two sigma surfaces (followed by top and bottom sigma values expressed in 1/100, separated by a space)	(layer sigma-between 99.5 100.0) for layer between .995 and 1.0

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
sigma-between-xp	Layer between two sigma levels (followed by top and bottom sigma values expressed as 1.1-sigma)	(layer sigma-between-xp .105 .100) for layer between .995 and 1.0
surface	Earth's surface	(layer surface)
theta	Isentropic (theta) level (followed by potential temperature in degrees K)	(layer theta 300)
theta-between	Layer between two isentropic surfaces (followed by top and bottom values expressed as 475-theta in degrees K)	(layer theta-between 150 200)
tropopause	Level of tropopause (top of troposphere)	(layer tropopause)

PowerPoint API Function Description Table

Method	Description	Example
Application	Represents the entire Microsoft PowerPoint application.	MyPath = Application .Path
ActivePresentation	Returns a Presentation object that represents the presentation open in the active window. (Read-only)	Application. ActivePresentation .Save As MyPath
Presentations	Returns a Presentation object that represents the presentation in which the specified document window or slide show window was created. (Read-only)	firstPresSlides = Windows(1).Presentation.Slides.Count Windows(2). Presentation .PageSetup _ .FirstSlideNumber = firstPresSlides + 1
Presentations.Add	Creates a presentation. Returns a Presentation object that represents the new presentation.	This example creates a presentation, adds a slide to it, and then saves the presentation. With Presentations.Add .Slides.Add 1, ppLayoutTitle .SaveAs "Sample" End With
Slides	A collection of all the Slide objects in the specified presentation.	Use the Slides property to return a Slides collection: ActivePresentation.Slides.Add 2, ppLayoutBlank

Slides.Add	Creates a new slide and adds it to the collection of slides in the specified presentation. Returns a Slide object that represents the new slide.	This example adds a blank slide at the end of the active presentation. With ActivePresentation.Slides .Add .Count + 1, ppLayoutBlank End With
Shapes	A collection of all the Shape objects on the specified slide. Each Shape object represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, or picture.	Use the Shapes property to return the Shapes collection. The following example selects all the shapes on myDocument. Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes.Select All
Shapes.AddPicture	Creates a picture from an existing file. Returns a Shape object that represents the new picture.	Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes. AddPicture "c:\microsoft office\" & _ "clipart\music.bmp", True, True, 100, 100, 70, 70

Shapes.PictureFormat	Contains properties and methods that apply to pictures and OLE objects. The LinkFormat object contains properties and methods that apply to linked OLE objects only. The OLEFormat object contains properties and methods that apply to OLE objects whether or not they're linked.	Set myDocument = ActivePresentation.Slides(1) With myDocument.Shapes(1).PictureFormat .Brightness = 0.3 .Contrast = 0.7 .ColorType = msoPictureGrayScale .CropBottom = 18 End With
SlideShowTransition	Contains information about how the specified slide advances during a slide show.	With ActivePresentation.Slides(1).SlideShowTransition .Speed = ppTransitionSpeedFast End With
SlideShowSetting	Represents the slide show setup for a presentation.	With ActivePresentation.SlideShowSettings .RangeType = ppShowSlideRange End With

THIS PAGE IS INTENTIONALLY LEFT BLANK

APPENDIX H. VISUAL BASIC IMPLEMENTATION

1. Configuration GUI (CBcfg)

```
VERSION 5.00
Begin VB.Form CBform
    BackColor      = &H80000004&
    Caption        = "CBcfg"
    ClientHeight   = 9195
    ClientLeft     = 60
    ClientTop      = 345
    ClientWidth    = 8490
    LinkTopic      = "Form1"
    ScaleHeight    = 9195
    ScaleWidth     = 8490
    StartUpPosition = 3 'Windows Default
Begin VB.TextBox VirtualDirText
    Height         = 375
    Left           = 1080
    TabIndex       = 3
    Tag            = "3"
    Top            = 7320
    Width          = 6375
End
Begin VB.TextBox TypeText
    Height         = 375
    Left           = 1080
    TabIndex       = 1
    Top            = 5160
    Width          = 6375
End
Begin VB.CommandButton Delete
    Caption        = "Delete"
    Enabled        = 0 'False
    BeginProperty Font
        Name        = "MS Sans Serif"
        Size        = 9.75
        Charset     = 0
        Weight      = 700
        Underline   = 0 'False
        Italic      = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height         = 375
    Left           = 4440
    TabIndex       = 6
    Top            = 8160
    Width          = 1335
End
Begin VB.CommandButton Add
    Caption        = "Set"
    Enabled        = 0 'False
    BeginProperty Font
        Name        = "MS Sans Serif"
        Size        = 9.75
        Charset     = 0
```

```

        Weight          = 700
        Underline        = 0      'False
        Italic           = 0      'False
        Strikethrough    = 0      'False
    EndProperty
    Height              = 375
    Left                = 6120
    TabIndex            = 7
    Top                 = 8160
    Width               = 1335
End
Begin VB.CommandButton Cancel
    Caption              = "Cancel"
    BeginProperty Font
        Name              = "MS Sans Serif"
        Size              = 9.75
        Charset           = 0
        Weight            = 700
        Underline         = 0      'False
        Italic            = 0      'False
        Strikethrough     = 0      'False
    EndProperty
    Height              = 375
    Left                = 2760
    TabIndex            = 5
    Top                 = 8160
    Width               = 1335
End
Begin VB.CommandButton OK
    Caption              = "OK"
    BeginProperty Font
        Name              = "MS Sans Serif"
        Size              = 9.75
        Charset           = 0
        Weight            = 700
        Underline         = 0      'False
        Italic            = 0      'False
        Strikethrough     = 0      'False
    EndProperty
    Height              = 375
    Left                = 1080
    TabIndex            = 4
    Top                 = 8160
    Width               = 1335
End
Begin VB.TextBox LocationText
    Height              = 375
    Left                = 1080
    TabIndex            = 2
    Tag                 = "3"
    Top                 = 6240
    Width               = 6375
End
Begin VB.ListBox dataList
    Height              = 3570
    Left                = 1080
    TabIndex            = 0
    Top                 = 720

```

```

        Width          = 6375
    End
    Begin VB.Label Label2
        Caption          = "Virtual directory (optional):"
        BeginProperty Font
            Name           = "MS Sans Serif"
            Size           = 9.75
            Charset        = 0
            Weight         = 700
            Underline      = 0 'False
            Italic         = 0 'False
            Strikethrough   = 0 'False
        EndProperty
        Height           = 255
        Left             = 1080
        TabIndex        = 11
        ToolTipText      = "A virtual directory associated
with the key used by the Web server."
        Top             = 6840
        Width           = 2775
    End
    Begin VB.Label Label4
        Caption          = "Key:"
        BeginProperty Font
            Name           = "MS Sans Serif"
            Size           = 9.75
            Charset        = 0
            Weight         = 700
            Underline      = 0 'False
            Italic         = 0 'False
            Strikethrough   = 0 'False
        EndProperty
        Height           = 255
        Left             = 1080
        TabIndex        = 10
        ToolTipText      = "An image type or any other
variable name."
        Top             = 4680
        Width           = 615
    End
    Begin VB.Label Label3
        Caption          = "Directory:"
        BeginProperty Font
            Name           = "MS Sans Serif"
            Size           = 9.75
            Charset        = 0
            Weight         = 700
            Underline      = 0 'False
            Italic         = 0 'False
            Strikethrough   = 0 'False
        EndProperty
        Height           = 255
        Left             = 1080
        TabIndex        = 9
        ToolTipText      = "An actual directory associated
with the key."
        Top             = 5760
        Width           = 1095
    End

```

```

End
Begin VB.Label Label1
    Caption           =   "Current configuration:"
    BeginProperty Font
        Name           =   "MS Sans Serif"
        Size           =   9.75
        Charset         =   0
        Weight          =   700
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
    EndProperty
    Height            =   255
    Left              =   1080
    TabIndex          =   8
    ToolTipText       =   "The current setting for
Continuous Brief application."
    Top               =   240
    Width             =   2295
End
End
Attribute VB_Name = "CBform"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'#####
'#
'#   File: CBform.frm
'#   Date           Author           Histor
'#   5/31/2000      Tam Tran         Created.
'#
'#   CBcfg is an utility application that provides a
'#   Graphical User Interface (GUI) for setting the image
'#   type and its location. This application supports the
'#   configuration of CBWrapper.
'#
'#####
'*****
',
'   String variables that hold the locations where to find
'   the configuration file (cbdata.cfg), and the temporary
'   directory for this application during run time.
',
'*****
Private cfgfile As String
Private cfgtmp As String
'*****
',
'   Unload the CBcfg form when the Cancel button is clicked.
',
'*****
Private Sub Cancel_Click()
    Unload Me
End Sub
'*****
',

```

```

' Display information for each record selected from the
' current configuration list box.
'
'*****
Private Sub dataList_Click()
    Dim listStr As String
    Dim typeStr As String
    Dim locationStr As String
    Dim virtualStr As String

    listStr = dataList.Text
    Call lineInfo(listStr, typeStr, locationStr,
virtualStr)
    ' Display the key name in the Key text box.
    TypeText.Text = typeStr
    ' Display the directory associated with the key in the
    ' Directory text box.
    LocationText.Text = locationStr
    ' Display the virtual directory associated with the key
    ' in the Virtual Directory text box
    VirtualDirText.Text = virtualStr
    Add.Enabled = False
    Delete.Enabled = True
End Sub
'*****
'
' Tasks done when deleting an item from the list.
' First, copy all lines from the cfgfile to the cfgtmp
' file except the line that's being deleted. Then copy
' back to the cfgfile from the cfgtmp.
'
'*****
Private Sub Delete_Click()
    Open cfgfile For Input As #1
    Open cfgtmp For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        If Not (InStr(1, inputStr, TypeText.Text & "=",
vbTextCompare) > 0) Then
            Print #2, inputStr
        End If
    Loop
    Close #1
    Close #2
    ' Copy the cfgtmp to the cfgfile
    Open cfgtmp For Input As #1
    Open cfgfile For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        Print #2, inputStr
    Loop
    Close #1
    Close #2
    Call updateList
End Sub
'*****
'
' Tasks done when the application is load.

```



```

' This requires two system environment variables set,
' which are CB_HOME, where the cbdata.cfg is located, and
' CB_TMP, where the temporary file is created.
'
'*****
Private Sub Form_Load()
    cfgfile = Environ("CB_HOME") & "\cbdata.cfg"
    cfgtmp = Environ("TEMP") & "\cbdata_.tmp"
    Call updateList
End Sub
'*****
'
' Activate the Add button if new value is entered from
' the Image type box.
'
'*****
Private Sub KeyText_Change()
    Add.Enabled = True
End Sub
'*****
'
' Save the changes (if any), and close the CBcfg form
' when the OK button is clicked
'
'*****
Private Sub OK_Click()
    If (Add.Enabled) Then
        Call Add_Click
    End If
    Unload Me
End Sub
'*****
'
' The lineInfo subroutine parses a line input from the
' configuration file (cbdata.cfg). It separates information
' of the key, the directory, and the virtual directory
' from the line string input.
' Parameters:
'     in:
'         searchStr - the string is being parsed.
'     in/out:
'         K - a variable that holds the key string
'         D - a variable that holds the directory string
'         V - a variable that holds the virtual directory
string
'
'*****
Private Sub lineInfo(searchStr As String, K As String, D As
String, V As String)
    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=", vbTextCompare)
    ' Get the key string
    K = Mid(searchStr, istart, istop - 1)
    istart = istop + 1
    istop = InStr(istart, searchStr, "|", vbTextCompare)
    ' Get the directory string

```

```

        If istop > istart Then
            D = Mid(searchStr, istart, istop - istart)
            istart = istop + 1
            'Get the location string
            V = Mid(searchStr, istart)
        Else
            D = Mid(searchStr, istart)
            V = ""
        End If
    End Sub
End Sub
'*****
'
' Tasks done when adding an item to the list. First, check
' if there is any line from cfgfile that has the same key
' value as the added item. Then update it with the new
' value. Otherwise, add a new line (item) to the cfgfile.
'
'*****
Private Sub Add_Click()
    Add.Enabled = False
    Open cfgfile For Input As #1
    Open cfgtmp For Output As #2
    ' Check for whether or not the image type exists.
    Do While Not EOF(1)
        Line Input #1, inputStr
        If Not (InStr(1, inputStr, TypeText.Text & "=",
vbTextCompare) > 0) Then
            ' Write to a temporary file
            Print #2, inputStr
        End If
    Loop
    If (StrComp("", VirtualDirText.Text, vbTextCompare) =
0) Then
        Print #2, TypeText.Text & "=" & LocationText.Text
    Else
        Print #2, TypeText.Text & "=" & LocationText.Text &
"|" & VirtualDirText.Text
    End If
    Close #1
    Close #2
    ' Copy the cfgtmp to the cfgfile
    Open cfgtmp For Input As #1
    Open cfgfile For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        Print #2, inputStr
    Loop
    Close #1
    Close #2
    Call updateList
End Sub
'*****
'
' Activate the Add button if new value is entered from
' the Key text box.
'
'*****
Private Sub TypeText_Change()

```

```

        Add.Enabled = True
End Sub
'*****
'
' Activate the Add button if new value is entered from
' the Directory text box.
'
'*****
Private Sub locationText_Change()
    Add.Enabled = True
End Sub
'*****
'
' Refresh the GUI after adding or deleting an item from
' the list.
'
'*****
Private Sub updateList()
    Dim intFile As Integer
    dataList.Clear

    intFile = FreeFile()
    Open cfgfile For Input As #intFile
    Do While Not EOF(intFile) ' Check for end of file.
        Line Input #intFile, inputStr ' Read line of data.
        dataList.AddItem inputStr
    Loop
    Close #intFile
    TypeText.Text = ""
    LocationText.Text = ""
    VirtualDirText.Text = ""
    Add.Enabled = False
    Delete.Enabled = False
End Sub
'*****
'
' Activate the Add button if new value is entered from
' the Virtual Directory text box.
'
'*****
Private Sub VirtualDirText_Change()
    Add.Enabled = True
End Sub

```

2. Application Wrapper (CBWrapper)

```

VERSION 5.00
Object = "{48E59290-9880-11CF-9754-00AA00C00908}#1.0#0";
"MSINET.OCX"
Begin VB.UserControl WebInterface
    BackColor      = &H80000001&
    ClientHeight   = 5475
    ClientLeft     = 0
    ClientTop      = 0
    ClientWidth    = 8430
    ScaleHeight    = 5475
    ScaleWidth     = 8430
    Begin InetCtrlsObjects.Inet Inet1

```

```

Left          = 120
Top           = 120
_ExtentX      = 1005
_ExtentY      = 1005
_Version      = 393216
End
Begin VB.TextBox ImagesText
BeginProperty Font
    Name          = "Arial"
    Size          = 9.75
    Charset       = 0
    Weight        = 700
    Underline     = 0    'False
    Italic        = 0    'False
    Strikethrough = 0    'False
EndProperty
Height        = 375
Left         = 5880
TabIndex     = 7
Text         = "24"
Top          = 1680
Width        = 735
End
Begin VB.TextBox HeightText
BeginProperty Font
    Name          = "Arial"
    Size          = 9.75
    Charset       = 0
    Weight        = 700
    Underline     = 0    'False
    Italic        = 0    'False
    Strikethrough = 0    'False
EndProperty
Height        = 375
Left         = 5880
TabIndex     = 6
Text         = "540"
Top          = 2520
Width        = 735
End
Begin VB.TextBox WidthText
BeginProperty Font
    Name          = "Arial"
    Size          = 9.75
    Charset       = 0
    Weight        = 700
    Underline     = 0    'False
    Italic        = 0    'False
    Strikethrough = 0    'False
EndProperty
Height        = 375
Left         = 5880
TabIndex     = 5
Text         = "720"
Top          = 3360
Width        = 735
End
Begin VB.TextBox DurationText

```

```

BeginProperty Font
    Name           =    "Arial"
    Size           =    9.75
    Charset        =    0
    Weight         =    700
    Underline      =    0    'False
    Italic         =    0    'False
    Strikethrough  =    0    'False
EndProperty
Height           =    375
Left            =    5880
TabIndex        =    4
Text            =    "0"
Top             =    4200
Width           =    735
End
Begin VB.CommandButton Start
    Caption       =    "Start"
    BeginProperty Font
        Name           =    "Arial"
        Size           =    9.75
        Charset        =    0
        Weight         =    700
        Underline      =    0    'False
        Italic         =    0    'False
        Strikethrough  =    0    'False
    EndProperty
    Height        =    495
    Left          =    720
    TabIndex      =    3
    Top           =    2400
    Width         =    1215
End
Begin VB.CommandButton Default
    Caption       =    "Default"
    BeginProperty Font
        Name           =    "Arial"
        Size           =    9.75
        Charset        =    0
        Weight         =    700
        Underline      =    0    'False
        Italic         =    0    'False
        Strikethrough  =    0    'False
    EndProperty
    Height        =    495
    Left          =    720
    TabIndex      =    2
    Top           =    4080
    Width         =    1215
End
Begin VB.ComboBox ImageType
    BeginProperty Font
        Name           =    "Arial"
        Size           =    9.75
        Charset        =    0
        Weight         =    700
        Underline      =    0    'False
        Italic         =    0    'False

```

```

        Strikethrough = 0 'False
    EndProperty
    Height = 360
    Left = 720
    TabIndex = 1
    Text = "Select an image type"
    Top = 1680
    Width = 2895
End
Begin VB.CommandButton Stop
    BackColor = &H00C0C0C0&
    Caption = "Stop"
    BeginProperty Font
        Name = "Arial"
        Size = 9.75
        Charset = 0
        Weight = 700
        Underline = 0 'False
        Italic = 0 'False
        Strikethrough = 0 'False
    EndProperty
    Height = 495
    Left = 720
    MaskColor = &H800000004&
    TabIndex = 0
    Top = 3240
    Width = 1215
End
Begin VB.Label images
    BackColor = &H800000001&
    Caption = "Images:"
    BeginProperty Font
        Name = "Arial"
        Size = 9.75
        Charset = 0
        Weight = 700
        Underline = 0 'False
        Italic = 0 'False
        Strikethrough = 0 'False
    EndProperty
    ForeColor = &H80000000E&
    Height = 255
    Left = 4800
    TabIndex = 14
    Top = 1680
    Width = 855
End
Begin VB.Label Label1
    BackColor = &H800000001&
    Caption = "Height:"
    BeginProperty Font
        Name = "Arial"
        Size = 9.75
        Charset = 0
        Weight = 700
        Underline = 0 'False
        Italic = 0 'False
        Strikethrough = 0 'False
    EndProperty

```

```

EndProperty
ForeColor      =    &H8000000E&
Height        =    255
Left          =    4800
TabIndex      =    13
Top           =    2520
Width         =    735
End
Begin VB.Label Label2
    BackColor    =    &H80000001&
    Caption      =    "Width:"
    BeginProperty Font
        Name      =    "Arial"
        Size      =    9.75
        Charset    =    0
        Weight     =    700
        Underline  =    0    'False
        Italic     =    0    'False
        Strikethrough =    0    'False
    EndProperty
    ForeColor    =    &H8000000E&
    Height       =    255
    Left        =    4800
    TabIndex    =    12
    Top         =    3360
    Width       =    735
End
Begin VB.Label Label3
    BackColor    =    &H80000001&
    Caption      =    "Duration:"
    BeginProperty Font
        Name      =    "Arial"
        Size      =    9.75
        Charset    =    0
        Weight     =    700
        Underline  =    0    'False
        Italic     =    0    'False
        Strikethrough =    0    'False
    EndProperty
    ForeColor    =    &H8000000E&
    Height       =    255
    Left        =    4800
    TabIndex    =    11
    Top         =    4200
    Width       =    855
End
Begin VB.Label Label4
    BackColor    =    &H80000001&
    Caption      =    "Second(s)"
    BeginProperty Font
        Name      =    "Arial"
        Size      =    9.75
        Charset    =    0
        Weight     =    700
        Underline  =    0    'False
        Italic     =    0    'False
        Strikethrough =    0    'False
    EndProperty

```

```

        ForeColor      =    &H80000000E&
        Height         =    255
        Left            =    6840
        TabIndex        =    10
        Top             =    4200
        Width           =    975
    End
    Begin VB.Label Label5
        Alignment        =    2    'Center
        BackColor        =    &H800000001&
        Caption          =    "CONTINUOUS BRIEF"
        BeginProperty Font
            Name          =    "MS Sans Serif"
            Size          =    18
            Charset       =    0
            Weight        =    700
            Underline     =    0    'False
            Italic        =    0    'False
            Strikethrough =    0    'False
        EndProperty
        ForeColor        =    &H80000000E&
        Height           =    495
        Left             =    2280
        TabIndex         =    9
        Top              =    360
        Width            =    3975
    End
    Begin VB.Label type
        BackColor        =    &H800000001&
        Caption          =    "Image type:"
        BeginProperty Font
            Name          =    "Arial"
            Size          =    9.75
            Charset       =    0
            Weight        =    700
            Underline     =    0    'False
            Italic        =    0    'False
            Strikethrough =    0    'False
        EndProperty
        ForeColor        =    &H80000000E&
        Height           =    255
        Left             =    720
        TabIndex         =    8
        Top              =    1200
        Width            =    1215
    End
End
Attribute VB_Name = "WebInterface"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#####
'# File: WebInterface.ctl
'# Date                Author                History
'# 5/31/2000            Tam Tran              Created.
'#####
Option Explicit

```



```

' *****
'
' The Continuous Brief wrapper (CBWrapper) is an ActiveX
' Control that represents the Graphical User Interface
' (GUI) via the Web browser (Internet Explorer). It allows
' an user to select the type of images that he/she wants
' to view. Also, it allows the user to set the number of
' images, the size, and the duration for the display.
'
' *****
Private mControllerConnector As ControllerConnector
Private mMonitor As Monitor
Private mMonitorConnector As MonitorConnector
Private WithEvents mController As Controller
Attribute mController.VB_VarHelpID = -1
' Get reference to Application object from the PowerPoint
API.
Public myPPT As PowerPoint.Application
Public AppRunning As Boolean
Private BriefStarted As Boolean
Private downloadFolder As String
Private cfgFolder As String
Private ServerURL As String

' *****
'
' Reset the Continuous Brief GUI to its default values.
' Set slide show to fullscreen size.
' Set number of images to 24
' Set duration of the slide show to 0.
'
' *****
Private Sub Default_Click()
    ImageType.Text = "Select an image type"
    ImagesText.Text = "24"
    HeightText.Text = "540"
    WidthText.Text = "720"
    DurationText.Text = "0"
End Sub

' *****
'
' Update the brief.
' Use the GetImageDir method from the Controller object
' to get the location of the files.
' Use the Controller_UpdateBrief method to update the
brief.
'
' *****
Private Sub Start_Click()
    Dim imageloc As String
    BriefStarted = True
    Call mController_UpdateBrief(ImageType.Text)
End Sub

' *****
'
' Stop the slide show.

```

```

' Terminate the background running PowerPoint application.
' Free up the un-used object.
' Reset the AppRunning flag to false.
'
'*****
Private Sub Stop_Click()
    If AppRunning Then
        myPPT.ActivePresentation.Close
        myPPT.Quit
        Set myPPT = Nothing
        AppRunning = False
        BriefStarted = False
    End If
End Sub

'*****
'
' Initialize references to the Monitor and Controller
objects.
'
'*****
Private Sub UserControl_Initialize()

    Set mControllerConnector = New ControllerConnector
    Set mController = mControllerConnector.Controller
    Set mMonitorConnector = New MonitorConnector
    Set mMonitor = mMonitorConnector.Monitor
    AppRunning = False
    BriefStarted = False

    ' Add image types to the drop-box in the Continuous
Brief GUI
    Dim intFile As Integer    ' FreeFile variable
    Dim inputStr As String
    Dim cfgFile As String
    Dim typeStr As String
    Dim locationStr As String
    Dim virtualDirStr As String
    Dim tmpFolderStr As String
    Dim tmpFileStr As String
    Dim downloadFileStr As String

    ' Set values for the URL, download folder, and a
temporary filename
    ' %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ' Change config here:
    ServerURL = "http://tampc.spawar.navy.mil/"
    ' %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    cfgFile = "cbdata.cfg"
    downloadFolder = Environ("TEMP") & "\cbdownload"
    cfgFolder = downloadFolder & "\cbdata"
    tmpFileStr = cfgFolder & "\" & cfgFile

    ' Download the "cbdata.cfg" file
    downloadFileStr = ServerURL & "/" & cfgFile

    ' Create a temporary directory for downloading data
    Call createFolder(downloadFolder)

```

```

Call createFolder(cfgFolder)
Call downloadFile(downloadFileStr, tmpFileStr)

intFile = FreeFile()
Open tmpFileStr For Input As #intFile
Do While Not EOF(intFile)
    Line Input #intFile, inputStr
    Call lineInfo(inputStr, typeStr, locationStr,
virtualDirStr)
    ImageType.AddItem typeStr
Loop
Close #intFile
End Sub

'*****
'
' Receive Controller event to do the update for the brief.
' Parameters:
'     in: DataType - the data (images) type
'     in: imageDir - the directory where to find the
images.
'*****
Private Sub mController_UpdateBrief(DataType As String)

    ' Check for the right type of data that the CBWrapper
is showing.
    If (StrComp(ImageType.Text, DataType, vbTextCompare) =
0) And BriefStarted Then
        Dim virtualDir As String
        Dim fileListName As String
        Dim tmpFileStr As String
        Dim tmpURLStr As String
        Call mController.GetImageInfo(ImageType.Text,
ImagesText.Text, _
                                virtualDir,
fileListName)

        ' Local variables declarations
        Dim myArray() As String
        Dim myPres As Presentation
        Dim fs, f, fc, fl, i, j, K
        Dim s As Slide
        Dim LeftVal As Long
        Dim TopVal As Long
        Dim imageW As Long
        Dim imageH As Long
        Dim ImgFile As String
        Dim intFile As Integer
        Dim inputStr As String

        ' Download the list of image filenames from server
        tmpURLStr = ServerURL & virtualDir &
"/CB_listfile/" & fileListName
        tmpFileStr = cfgFolder & "\" & fileListName
        Call downloadFile(tmpURLStr, tmpFileStr)

        ' Download image files from server
        intFile = FreeFile()

```

```

Open tmpFileStr For Input As #intFile
Do While Not EOF(intFile)
    Line Input #intFile, inputStr
    tmpURLStr = ServerURL & virtualDir & "/" &
inputStr
    tmpFileStr = downloadFolder & "\" & inputStr
    Call downloadFile(tmpURLStr, tmpFileStr)
Loop
Close #intFile

' Get reference to the PowerPoint Application
object.
On Error Resume Next
Set myPPT = GetObject(, "PowerPoint.application")
If Err.Number <> 0 Then
    Set myPPT =
CreateObject("PowerPoint.application")
End If

' Set the AppRunning flag so that it will be
' checked when the STOP button is clicked.
AppRunning = True

' Stop the current running slide show (if any)
If myPPT.Presentations.Count <> 0 Then
    myPPT.ActivePresentation.Close
End If

' Create new presentation with the new update data
Set myPres = myPPT.Presentations.Add(True)

' Create a FileSystemObject for manipulating the
file system
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(downloadFolder)
Set fc = f.Files
i = 1
K = 1

' Store all filenames from the image directory
' to an array for sorting purpose.
ReDim myArray(1 To fc.Count)
For Each f1 In fc
    myArray(i) = f1.Name
    i = i + 1
Next
' Sort the array.
Call mMonitor.dhBubbleSort(myArray)

' Calculate the positions and dimensions for the
images.
Call GetDimensions(LeftVal, TopVal, imageW, imageH)

' Add the images to the PowerPoint presentation.
For j = (fc.Count - ImagesText.Text + 1) To
fc.Count
    ImgFile = downloadFolder & "\" & myArray(j)
    myPres.Slides.Add K, ppLayoutBlank

```

```

        myPres.Slides.Item(K).Shapes.AddPicture
ImgFile, True, True, _
LeftVal, TopVal, imageW, imageH
        K = K + 1
    Next
    'Free up the FileSystemObject when done
    Set fs = Nothing
    Set f = Nothing
    Set fc = Nothing

    ' Configure the slide show properties and run the
show
    For Each s In myPPT.ActivePresentation.Slides
        With s.SlideShowTransition
            .AdvanceOnTime = True
            .AdvanceTime = DurationText.Text
        End With
    Next

    With myPPT.ActivePresentation.SlideShowSettings
        .StartingSlide = 1
        .EndingSlide = ImagesText.Text
        .AdvanceMode = ppSlideShowUseSlideTimings
        .LoopUntilStopped = True
        .Run
    End With

    ' Delete the images when done creating the brief
    For i = 1 To fc.Count
        If fs.FileExists(downloadFolder & "\" & myArray(i))
Then
            Set f = fs.DeleteFile(downloadFolder & "\" &
myArray(i), True)
        End If
    Next
    End If
End Sub

' *****
'
' The GetDimensions subroutine calculates the positions
' (Left, Top), and the dimensions (Height, Width)
' for the images.
' Parameters:
'     in/out: L - the Left value
'             T - the Top value
'             W - the Width value
'             H - the Height value
' *****

Private Sub GetDimensions(L As Long, T As Long, W As Long,
H As Long)

    ' Local variables declarations
    Dim DeltaX As Long
    Dim DeltaY As Long

```

```

        DeltaX = myPPT.ActivePresentation.PageSetup.SlideWidth
- WidthText.Text
        DeltaY = myPPT.ActivePresentation.PageSetup.SlideHeight
- HeightText.Text
        If DeltaX <= 0 Then
            L = 0
        Else
            L = DeltaX / 2
        End If
        If DeltaY <= 0 Then
            T = 0
        Else
            T = DeltaY / 2
        End If
        W = WidthText.Text
        H = HeightText.Text
        If W > 720 Then W = 720
        If H > 540 Then H = 540
    End Sub
'*****
'
' The lineInfo subroutine parses a line input from the
' configuration file (cbdata.cfg). It separates information
' of the key, the directory, and the virtual directory
' from the line string input.
' Parameters:
'     in:
'         searchStr - the string is being parsed.
'     in/out:
'         K - a variable that holds the key string
'         D - a variable that holds the directory string
'         V - a variable that holds the virtual directory
string
'*****
Private Sub lineInfo(searchStr As String, K As String, D As
String, V As String)
    Dim istart As Integer
    Dim istop As Integer
    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=", vbTextCompare)
    ' Get the key string
    K = Mid(searchStr, istart, istop - 1)
    istart = istop + 1
    istop = InStr(istart, searchStr, "|", vbTextCompare)
    ' Get the directory string
    If istop > istart Then
        D = Mid(searchStr, istart, istop - istart)
        istart = istop + 1
        'Get the location string
        V = Mid(searchStr, istart)
    Else
        D = Mid(searchStr, istart)
        V = ""
    End If
End Sub
'*****

```

```

'
' The downloadFile subroutine uses the OpenURL method to
' download a file from the current open connection using
' HTTP protocol.
' Parameters:
'     in:
'         URLStr - the URL for download the file from.
'         saveFile - the filename for storing the
'                   downloaded file on the client machine.
'
'*****
Private Sub downloadFile(URLStr As String, saveFile As
String)
    Dim bData() As Byte        ' Data variable
    Dim intFile As Integer     ' FreeFile variable
    intFile = FreeFile()      ' Set intFile to an unused
file.

    ' The result of the OpenURL method goes into the Byte
    ' array, and the Byte array is then saved to disk.
    bData() = Inet1.OpenURL(URLStr, icByteArray)
    Open saveFile For Binary Access Write As #intFile
    Put #intFile, , bData()
    Close #intFile
End Sub
'*****
' Creating a folder on client machine.
' Parameter:
'     in: path - a qualify name of the folder being
created.
'
'*****
Private Sub createFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If Not fs.FolderExists(path) Then
        Set f = fs.createFolder(path)
    End If
    Set fs = Nothing
    Set f = Nothing
End Sub
'*****
' Deleting a folder on a client machine.
' Parameter:
'     in: path - a qualify name of the folder being
deleted.
'
'*****
Private Sub deleteFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If fs.FolderExists(path) Then
        fs.deleteFolder path, True
    End If
    Set fs = Nothing
End Sub

```

```

'*****
'
' Clean up all temporary folder created when exiting.
'
'*****
Private Sub UserControl_Terminate()
    ' Delete the download folder
    deleteFolder downloadFolder
End sub

```

3. Object Components (Continuous Brief)

a) Global Variable Declarations

```

Attribute VB_Name = "GlobalDeclarations"
'#####
###
'# File: GlobalDeclarations.bas
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####
###
Option Explicit
'*****
***
'
' The cfgInfo type is a record that stores the
information
' that read from the cvdata.cfg file (i.e., Key,
Directory,
' Virtual Directory, and the stamped date, which is
the last
' time the data is checked.)
'
'*****
***
Public Type cfgInfo
    key As String
    path As String
    vir_path As String
    stampdate As Date
End Type

'*****
***
'
'Global variables used by the ControllerConnector
'
'*****
***

Public gController As Controller ' Reference to
controller object
Public gControllerUseCount As Long ' Global reference
count

```



```

'*****
***
'
' Global variables used by the MonitorConnector
'
'*****
***
Public gMonitor As Monitor           'Reference to
monitor object
Public gMonitorUseCount As Long      ' Global reference
count

'*****
***
'
' Global variables used by the Monitor and Controller
objects.
'
'*****
***
Public gCfgArray() As cfgInfo
b) Timer

VERSION 5.00
Begin VB.Form Timing
Caption = "Form1"
ClientHeight = 3195
ClientLeft = 60
ClientTop = 345
ClientWidth = 4680
LinkTopic = "Form1"
ScaleHeight = 3195
ScaleWidth = 4680
StartupPosition = 3 'Windows Default
Begin VB.Timer Clock
Left = 2160
Top = 1200
End
End
Attribute VB_Name = "Timing"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'#####
###
'# File: Timing.frm
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####
###
'*****
***
'
' Set the clock interval to 5 second.
' The Monitor component uses this timer event to poll
the
' storage directory for new data (images).

```

```

'
' *****
***
Private Sub Form_Load()
    Clock.Interval = 5000
End Sub

c) Controller

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
    Persistable = 0 'NotPersistable
    DataBindingBehavior = 0 'vbNone
    DataSourceBehavior = 0 'vbNone
    MTSTransactionMode = 0 'NotAnMTSObject
END
Attribute VB_Name = "Controller"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#####
###
'# File: Controller.cls
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####
###

Option Explicit

' *****
***
'
' The Controller component uses this UpdateBrief event
to
' notify the Continuous Brief wrapper (CBWrapper) for
' updating the brief.
' Event's parameters:
'     imageType: the type of images
'     imageLoc: the location where to find the
images.
'
' The Glue component will raise the event to notify
the
' Controller when it's done with storing data.
'
' The Monitor component will raise the event to notify
the
' Controller when the new data come in.
' WithEvents causes the component(s) which raise the
event(s)
' to run asynchronously.
' MonitorConnector component allows multiple
connections to
' single Monitor object.
'

```

```

'*****~
***
Event UpdateBrief(imageType As String)

Public WithEvents mGlue As Glue
Attribute mGlue.VB_VarHelpID = -1
Private WithEvents mMonitor As Monitor ' Get Monitor
events
Attribute mMonitor.VB_VarHelpID = -1
Private mMonitorConnector As MonitorConnector

'*****
***
'
' Connect to the Monitor component
'
'*****
***
Private Sub Class_Initialize()

    Set mMonitorConnector = New MonitorConnector
    Set mMonitor = mMonitorConnector.Monitor

End Sub

'*****
***
'
' Receive the notification from the Monitor component
' The Controller passes the information to the Glue
component
' for storing data to the database.
' Event's parameter:
'     DataType: the data (images) type
'
'*****
***
Private Sub mMonitor_NewData(DataType As String)
    Set mGlue = New Glue
    Call mGlue.StoreData(DataType)
End Sub

'*****
***
'
' Receive the notification from the Glue component
that
' Asynchronous glue component is done.
' The Controller notifies the CBWrapper(s) and passes
the
' information for the wrapper(s) to update the
brief(s).
' Event's parameter:
'     DataType: the data (images) type
'
'*****
***
Private Sub mGlue_GlueDone(DataType As String)

```

```

        Set mGlue = Nothing      ' Free the Glue object

        ' Notify the CBWrapper for updating the brief
        RaiseEvent UpdateBrief(DataType)
    End Sub
    '*****
***
    '
    ' Get all the image's filenames, which is being
    requested
    ' from the CBWrapper, and make the makeFileList
    function
    ' call to store the filenames to the CB_DATA.LST file.
    ' Parameters:
    '     in:
    '         ImageID - the image type
    '         fileCounts - the number of images
    requested.
    '         virtualDir - the virtual directory
    associated
    '                     with the images' directory.
    '     in/out:
    '         fileListName - a variable that holds the
    filename,
    '                     which contains the list of images'
    filenames.
    '
    '*****
***
    Public Sub GetImageInfo(ImageID As String, fileCounts
    As Integer, _
                                virtualDir As String,
    fileListName As String)

        Dim i As Integer
        For i = 1 To UBound(gCfgArray)
            If (StrComp(ImageID, gCfgArray(i).key,
vbTextCompare) = 0) Then
                virtualDir = gCfgArray(i).vir_path
                fileListName = "CB_DATA.LST"
                Call makeFileList(fileCounts,
gCfgArray(i).path, fileListName)
            End If
        Next
    End Sub
    '*****
***
    '
    ' Write all filenames from a specified directory to a
    file.
    ' This subroutine is called by GetImageInfo()
    ' Parameters:
    '     in:
    '         fileCounts - number of files is being
    read.
    '         path - a specified directory for getting
    the filenames.

```

```

'          filename - the file used for storing the
filenames.
'
' *****
***
Private Sub makeFileList(fileCounts As Integer, path
As String, _
                                filename As
String)
    Dim fs, f, fc, fl, i, j, a
    Dim myCount As Integer
    Dim listfileStr As String
    Dim myArray() As String

    ' Create a FileSystemObject for manipulating the
file system.
    Set fs =
CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(path)
    Set fc = f.Files
    myCount = fc.Count
    i = 1

    ' Store the name of the files to an array for
sorting purpose
    ReDim myArray(1 To myCount)
    For Each fl In fc
        myArray(i) = fl.Name
        i = i + 1
    Next

    ' Sort the array
    Call mMonitor.dhBubbleSort(myArray)
    listfileStr = path & "\" & "CB_listfile"
    createFolder listfileStr
    Set a = fs.CreateTextFile(listfileStr & "\" &
filename, True)
    For j = (myCount - fileCounts + 1) To myCount
        a.WriteLine (myArray(j))
    Next
    a.Close
    ' Free up the objects, which are no longer be
used.
    Set fs = Nothing
    Set f = Nothing
    Set fc = Nothing
    Set a = Nothing
End Sub
' *****
***
' This createFolder is used for creating a specified
folder.
' Parameter:
' in: path - the qualified name of the folder
being created.

```

```

'*****
***
Private Sub createFolder(path As String)
    Dim fs, f
    Set fs =
CreateObject("Scripting.FileSystemObject")
    If Not fs.FolderExists(path) Then
        Set f = fs.createFolder(path)
    End If
    Set fs = Nothing
    Set f = Nothing
End Sub

```

d) Controller Connector

```

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
    Persistable = 0 'NotPersistable
    DataBindingBehavior = 0 'vbNone
    DataSourceBehavior = 0 'vbNone
    MTSTransactionMode = 0 'NotAnMTSObject
END
Attribute VB_Name = "ControllerConnector"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#####
###
'# File: ControllerConnector.cls
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####
###

Option Explicit

'*****
***
' This property allows other components to get
reference
' to the Controller object.
'
'*****
***
Public Property Get Controller() As Controller
    Set Controller = gController
End Property

'*****
***
' Initilize Controller and reference count.
'
'*****
***
Private Sub Class_Initialize()

```

```

        If gController Is Nothing Then
            Set gController = New Controller
        End If
        gControllerUseCount = gControllerUseCount + 1
    End Sub

'*****
***
'
' Terminate controller when reference count = 0
'
'*****
***
Private Sub Class_Terminate()
    gControllerUseCount = gControllerUseCount - 1
    If gControllerUseCount = 0 Then
        'Set gList = Nothing
        Set gController = Nothing
    End If
End Sub
e) Monitor

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
    Persistable = 0 'NotPersistable
    DataBindingBehavior = 0 'vbNone
    DataSourceBehavior = 0 'vbNone
    MTSTransactionMode = 0 'NotAnMTSObject
END
Attribute VB_Name = "Monitor"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#####
###
'# File: Monitor.cls
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####
###
Option Explicit
'*****
***
' The VISStamDate, IRStampDate, and VAPORStampDate
variables
' store the created date of the latest stored data.
'
' WithEvents causes the component(s) which raise the
event(s)
' to run asynchronously.
' Event's parameter:
'     DataType: the data (images) type
'
' The Monitor component will raise the event to notify
the
' Controller when the new data come in.

```

```

'*****
***
Private VISStampDate As Date
Private IRStampDate As Date
Private VAPORStampDate As Date

Private mTiming As Timing
Private WithEvents mClock As Timer
Attribute mClock.VB_VarHelpID = -1

Event NewData(DataType As String)
'*****
***
'
' The tasks done when a new Monitor object is created.
'
'*****
***
Private Sub Class_Initialize()

' Start Monitor Timer and create instance of form
Set mTiming = New Timing
Load mTiming

' Connect timers' events to associated event
procedures in Monitor
Set mClock = mTiming.Clock

' Get the config information from the
configuration file
Call GetConfig
End Sub

'*****
***
'
' The tasks done when the Monitor object is
terminated.
'
'*****
***
Private Sub Class_Terminate() ' Terminate Monitor

' Free up the timer object.
Set mClock = Nothing

' Unload and free up the form.
Unload mTiming
Set mTiming = Nothing
End Sub

'*****
***
'
' Process Timer Event.
' This timer event causes the Monitor to poll the
storage
' directories for new data.

```



```

' The Monitor will raise the event(s) if it found a
new data.
'
' *****
***
Private Sub mClock_Timer()
    Dim i As Integer
    For i = 1 To UBound(gCfgArray)
        If IsNewFile(gCfgArray(i).path, i) Then
            RaiseEvent NewData(gCfgArray(i).key)
        End If
    Next
End Sub

' *****
***
' The IsNewFile function is used to determine whether
or
' not a new data exists.
' Parameters:
'     in: StrDir - the directory where to check for
'           new data.
'     in: StampDate - the created date of the latest
'           data from the previous
checked.
' Return:
'     TRUE if there's new data, and FALSE otherwise.
' *****
***
Private Function IsNewFile(StrDir As String,
arrayIndex As Integer) As Boolean
    ' Local variables declarations.
    Dim fs, f, fc, fl, i
    Dim myStamp As Date
    Dim myArray() As String

    ' Create a FileSystemObject for manipulating the
file system.
    Set fs =
CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(StrDir)
    Set fc = f.Files
    i = 1

    ' Store the name of the files to an array for
sorting purpose
    ReDim myArray(1 To fc.Count)
    For Each fl In fc
        myArray(i) = fl.Name
        i = i + 1
    Next

    ' Sort the array
    Call dhBubbleSort(myArray)

```

```

        ' Check for new file based on the file's created
date.
        myStamp = fs.GetFile(StrDir & "\" &
myArray(fc.Count)).DateCreated
        If (DateDiff("s", gCfgArray(arrayIndex).stampdate,
myStamp) <> 0) Then
            gCfgArray(arrayIndex).stampdate = myStamp
            IsNewFile = True
        Else
            IsNewFile = False
        End If

        ' Free up the objects, which are no longer be
used.
        Set fs = Nothing
        Set f = Nothing
        Set fc = Nothing
    End Function

    *****
***
    ' Standard bubblesort.
    ' DON'T USE THIS unless you know the data is already
    ' almost sorted! It's incredibly slow for
    ' randomly sorted data.

    ' There are many variants on this algorithm.
    ' There may even be better ones than this.
    ' But it's not even going to win any
    ' speed prizes for random sorts.

    ' From "Visual Basic Language Developer's Handbook"
    ' by Ken Getz and Mike Gilbert
    ' Copyright 2000; Sybex, Inc. All rights reserved.

    ' In:
    '   varItems:
    '       Array of items to be sorted.
    ' Out:
    '   VarItems will be sorted.
    *****
***
Public Sub dhBubbleSort(varItems As Variant)

    Dim blnSorted As Boolean
    Dim lngI As Long
    Dim lngJ As Long
    Dim lngItems As Long
    Dim varTemp As Variant
    Dim lngLBound As Long

    lngItems = UBound(varItems)
    lngLBound = LBound(varItems)

    ' Set lngI one lower than the lower bound.
    lngI = lngLBound - 1
    Do While (lngI < lngItems) And Not blnSorted
        blnSorted = True

```

```

        lngI = lngI + 1
        For lngJ = lngLBound To lngItems - lngI
            If varItems(lngJ) > varItems(lngJ + 1)
Then
                varTemp = varItems(lngJ)
                varItems(lngJ) = varItems(lngJ + 1)
                varItems(lngJ + 1) = varTemp
                blnSorted = False
            End If
        Next lngJ
    Loop
End Sub
'*****
***
'
' The lineInfo subroutine parses a line input from the
' configuration file (cbdata.cfg). It separates
information
' of the key, the directory, and the virtual directory
' from the line string input.
' Parameters:
'   in:
'       searchStr - the string is being parsed.
'   in/out:
'       K - a variable that holds the key string
'       D - a variable that holds the directory
string
'       V - a variable that holds the virtual
directory string
'
'*****
***
Private Sub lineInfo(searchStr As String, K As String,
D As String, V As String)
    Dim istart As Integer
    Dim istop As Integer

    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=",
vbTextCompare)
    ' Get the key string
    K = Mid(searchStr, istart, istop - 1)
    istart = istop + 1
    istop = InStr(istart, searchStr, "|",
vbTextCompare)
    ' Get the directory string
    If istop > istart Then
        D = Mid(searchStr, istart, istop - istart)
        istart = istop + 1
        'Get the location string
        V = Mid(searchStr, istart)
    Else
        D = Mid(searchStr, istart)
        V = ""
    End If
End Sub
End Sub

```

```

    '*****
***
    '
    ' The GetDateArrayIndex function returns an index of
the
    ' dateArray, where the specified image type (ID) is
stored.
    '
    '*****
***
    Public Function GetArrayIndex(key As String) As
Integer
        Dim tmpInfo As cfgInfo
        Dim bFound As Boolean
        Dim i As Integer
        bFound = False
        i = 1
        Do While Not bFound
            tmpInfo = gCfgArray(i)
            If (StrComp(tmpInfo.key, key) = 0) Then
                GetArrayIndex = i
                bFound = True
            End If
            i = i + 1
        Loop
    End Function
    '*****
***
    '
    ' The GetConfig subroutine reads information stored in
    ' the configuration file, and adds them to the link
list.
    '
    '*****
***
    Private Sub GetConfig()

        Dim cfgpath As String
        Dim inputStr As String
        Dim keyStr As String
        Dim dirStr As String
        Dim virDirStr As String
        Dim intFile As Integer
        Dim tmpInfo As cfgInfo

        ' Initialize the size the gCfgArray
        ReDim gCfgArray(0)
        ' Get the path for the configuration file
        cfgpath = Environ("CB_HOME") & "\cbdata.cfg"

        ' Store the configured info to the array
        intFile = FreeFile()
        Open cfgpath For Input As #intFile
        Do While Not EOF(intFile)
            Line Input #intFile, inputStr
            Call lineInfo(inputStr, keyStr, dirStr,
virDirStr)
            With tmpInfo

```

```

        .key = keyStr
        .path = dirStr
        .vir_path = virDirStr
        .stampdate = -1      ' initialize the date
to before Dec. 30, 1899
        End With
        ReDim Preserve gCfgArray(UBound(gCfgArray) +
1)
        gCfgArray(UBound(gCfgArray)) = tmpInfo
    Loop
    Close #intFile
End Sub

```

f) Monitor Connector

```

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1    'True
    Persistable = 0  'NotPersistable
    DataBindingBehavior = 0   'vbNone
    DataSourceBehavior  = 0   'vbNone
    MTSTransactionMode  = 0   'NotAnMTSObject
END
Attribute VB_Name = "MonitorConnector"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#####

###
'#  File: MonitorConnector.cls
'#  Date                Author                History
'#  5/31/2000            Tam Tran              Created.
'#####

###

Option Explicit
'*****
***
'
'   This property allows other components to get
reference
'   to the Monitor object.
'
'*****
***
Public Property Get Monitor() As Monitor
    Set Monitor = gMonitor
End Property
'*****
***
'
'   Initialize Monitor and reference count.
'
'*****
***
Private Sub Class_Initialize()
    If gMonitor Is Nothing Then

```

```

' Creates a new link list for holding the
configuration info.

```

```

    Set gMonitor = New Monitor
    End If
    gMonitorUseCount = gMonitorUseCount + 1
End Sub

```

```

'*****

```

```

***

```

```

'
' Terminate Monitor when reference count = 0
'

```

```

'*****

```

```

***

```

```

Private Sub Class_Terminate()
    gMonitorUseCount = gMonitorUseCount - 1
    If gMonitorUseCount = 0 Then
        Set gMonitor = Nothing
    End If
End Sub

```

g) Glue

```

VERSION 1.0 CLASS

```

```

BEGIN

```

```

    MultiUse = -1 'True
    Persistable = 0 'NotPersistable
    DataBindingBehavior = 0 'vbNone
    DataSourceBehavior = 0 'vbNone
    MTSTransactionMode = 0 'NotAnMTSObject

```

```

END

```

```

Attribute VB_Name = "Glue"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True

```

```

'#####

```

```

###

```

```

'# File: Glue.cls
'# Date Author History
'# 5/31/2000 Tam Tran Created.
'#####

```

```

###

```

```

Option Explicit

```

```

'*****

```

```

***

```

```

'
' The Glue component uses this event to notify the
' Controller when done with its task.
' Event's parameter:
'     DataType: the data (images) type.
'
'*****

```

```

***

```

```

Event GlueDone(DataType As String)

```

```

'*****

```

```

***

```

```

'

```

```

        ' Notify the Controller when done storing data.
        ' *****
***
Public Sub StoreData(DataType As String) ' Start glue
task
    ' <Insert glue task here>
    ' ...
    RaiseEvent GlueDone(DataType)
End Sub

```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218
2. Dudley Knox Library2
Naval Postgraduate School
411 Dyer Road
Monterey, California 93943-5101
3. Chairman, Code CS1
Naval Postgraduate School
Monterey, California 93943-5118
4. Dr. Luqi, CS/Lq1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118
5. Dr. Valdis Berzins, CS/Be1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118
6. Dr. Mantak Shing, CS/Sh1
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118
7. Tam M. Tran.1
SPAWAR SYS CEN, San Diego
53140 Systems St.
San Diego, CA 92152-7555
8. James O. Allen.1
SPAWAR SYS CEN, San Diego
53140 Systems St.
San Diego, CA 92152-7555